DOCUMENT RESUME

ED 169 947 IR 007 234

AUTHOR Cheng, Y. P.; And Others
TITLE CS 237--Fortran for Business. Course Guide.
INSTITUTION Nebraska Univ., Lincoln. Dept. of Computer Science.
PUB DATE Jan 79
NOTE 77p.; Not available in hard copy due to marginal print quality

EDRS PRICE MF01 Plus Postage. PC Not Available from EDRS.
DESCRIPTORS Business; *Business Education; Computer Science Education; Higher Education; Individualized Instruction; *Instructional Materials; *Programing

ABSTRACT

This guide was prepared for a lectureless two-semester-credit introductory programming course for business students at the University of Nebraska. The guide lays out course objectives, course mechanics, exercises and programming assignments, and describes the SIMPLE program editor and job submission system in use at the university. However, students must rely on a commercially published textbook for information about Fortran syntax. (Author/JVP)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

-ED169947

Name_____

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
NATIONAL INSTITUTE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

CS 237 - FORTRAN FOR BUSINESS

COURSE GUIDE

BEST COPY AVAILABLE

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

George Nagy

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC) AND USERS OF THE ERIC SYSTEM."

Staff:

Y.F. Chang
C. Daly
D. Embley
G. Nagy
W. Prange
A. Sulek

© 1979

Department of Computer Science
University of Nebraska-Lincoln

IR007234

January, 1979

$5.00

PREFACE

This course guide was prepared for lectureless CS 237, Fortran for Business. CS 237 is a two-credit course required for business majors taken in conjunction with MGM 237, Introduction to Computers for Business Decisions.

The substitution of other learning resources for traditional lectures represents an attempt to improve instruction in the introductory programming courses. The current experiment comprises eleven sections of 30-35 students each. Other potential candidates for improvement include over thirty sections per semester of introductory programming classes on the Lincoln Campus alone.

Initial encouragement and sponsorship for the experiment came from the UNL Teaching and Learning Council, which voted in February 1978 to fund the initial stage of development. This funding and encouragement was matched by the Dean of the College of Arts and Sciences, and two five-week research assistantships were also provided by the Summer Sessions Office. Finally, in August 1978, the Academic Services Group of the University of Nebraska Computing Network authorized a preliminary study to determine the feasibility of making the computer interface developed for this project, SIMPLE, available to all University computer users.

During the Fall Semester of the 1978-79 school year, this approach was followed in four sections of CS 237 while the remaining five sections received the traditional lecture oriented form of the course. Preliminary results were most encouraging, and it was decided to teach all sections of CS 237 without lectures in the Spring Semester.

We are most grateful to the numerous faculty and student in several colleges at both UNL and UNO who offered helpful suggestions on earlier versions of various componenets of the project. In particular, we appreciate the help of Dr. Tom Thompson and Steve Henderson of AGNET, of Len Campbell and Duane Tiemann of UNCN, and the comments of Professors Gerald Boardman and Del Wright (Teachers College), Richard Schonberger (CBA), and Charles Combs (UNO, Math and CS).

Finally, we thank the staff of the UN Computing Network and the students of the Department of Computer Science. Much of the programming on SIMPLE was capably performed by Len Campbell, Curt Hill, Julia Middleton, and

John Woodward. The account supervisor was the responsibility of Cindy Eby and Tom Lee, and Debbie Brong was in charge of the student tracking subsystem. Margert Hazen designed a course evaluation technique and is currently analyzing the results. The programming assignments were written by Essi Hassani and Norman Smith. LuAnn Murray typed and retyped and retyped the Course Guide as it was revised.

We are eager to hear from readers of this Guide, both students and others, with regard to what we are attempting to accomplish in this experiment.

Department of Computer Science
University of Nebraska-Lincoln
Lincoln, Nebraska 68588

Y.F. Chang, Associate Professor
Cecilia Daly, Instructor
David W. Embley, Assistant Professor
George Nagy, Professor and Chairman
William Prange, Instructor
A. Sulek, Instructor

January, 1979

TABLE OF CONTENTS

	Page
INTRODUCTION	1
COURSE OUTLINE AND OBJECTIVES	3
Course Objectives	3
Course Outline and Weekly Objectives	3
Scheduled Events	7
COURSE MECHANICS	9
How to Successfully Complete the Course	9
Grading	11
RESOURCES	13
The Text	13
The Computer	13
Course Headquarters	14
Instructors	14
Teaching Assistants	15
Debug Consultants	16
I/O Clerks	16
Other Students	16
HOW TO USE THE TERMINALS	17
Terminal Locations and Hours	17
Logging On and Off	17
Etiquette	19
Computer Accounts	20
SIMPLE	21
Introduction	21
Some Necessary Explanation	21
Some Elementary Commands	23
Executing a Program	25
Locate and Change	29
Working with More than One File	33
Add-Only Files	38
Summary of Commands	42
APPENDIX I - Follow-up Courses	43
APPENDIX II - IBM OS VM/370-158 CP-CMS	44
ASSIGNMENTS	47

WARNING: The students* have determined that CS 237 is hazardous to your free time.

*A survey of 100 students who took CS 237 in Fall '7. showed that they spent an average of 8.4 hours per week working on this course.

INTRODUCTION

In CS 237 you will learn to use the FORTRAN programming language to direct the computer to do something useful. Skilled programmers can use computers to perform all kinds of imaginable (and even unimaginable) computations and data manipulations. They can command the behavior of the computer because they have thoroughly mastered, mostly by practice and experience, the programming languages with which they work. FORTRAN is one of many programming languages, and if you learn to use it properly you too can tap the potential of the vast computational resources available to you. Even if you yourself don't ever program again, you will be able to interact intelligently with computer professionals.

Although it is easy to say what you should learn in CS 237, it is difficult to say how you should learn it. By observation, it seems that students learn to program best from direct computer feedback on programs submitted for execution, from carefully worked-out examples, and from individual, one-on-one interactions with each other, with teaching assistants, and with instructors. Furthermore, students seem to learn less in a traditional lecture setting than might be expected. Accordingly, for this course there are only a few formal lectures. Instead, you will meet more often and more formally with instructors and teaching assistants, use a text that contains an abundance of worked-out examples, and interact more often with the computer.

The teaching method is experimental, but you should find the class enjoyable and productive in terms of the time spent for what you learn.

If there are no lectures to attend, what do you do instead? This Course Guide tells you. It defines the course objectives, specifies the course mechanics, and explains how to take advantage of the learning resources available to you. Find a comfortable corner, sit down preferably with your toes above eye level, and read the Guide to the end. After about two weeks, read it again.

Briefly, you have three major resources:

1. course headquarters
2. the textbook, and
3. access to the computer through SIMPLE.

Course headquarters, Room 17 in Ferguson Hall, will be staffed by teaching assistants and instructors on a regular basis. The textbook chosen for the course is self-instruction oriented and contains numerous examples and illustrations to help you understand FORTRAN.

Accessing the computer is fundamental to the course since you can't learn to program without a lot of actual practice and experience. Sometimes just trying to communicate with the computer to develop and execute a program is a stumbling block that prevents many from really learning to program in FORTRAN. Hence a simple access method, called SIMPLE, has been devised specifically for use in this course.

With these resources available to you, you can learn FORTRAN, but you must take the initiative to do it. As with any course at the university, the instructor cannot make you learn; but can only provide you with the opportunity.

COURSE OUTLINE & OBJECTIVES

Course Objectives

Computer Science 237 is directed mainly to business students who are also taking Management 237, but anyone who wants to learn FORTRAN with business applications in mind is welcome to take these courses. The specific objectives of this course are:

1. to learn how to read and write FORTRAN-language statements,
2. to learn how to write and run FORTRAN programs, and
3. to learn about computers, computer-related equipment, and program-development procedures.

About 30% of your time will be devoted to learning how to read and write FORTRAN statements. To master FORTRAN, you have to know exactly where to place the commas and quote marks, the precise meaning of the key words, and much more. This activity is similar to learning vocabulary and proper grammar and punctuation.

About 60% of your time will be devoted to learning how to write and run FORTRAN programs. If you want the computer to do something useful, you have to tell it, statement by statement, exactly what to do and how to do it. This is analogous to composition.

The remaining 10% of your time will be spent learning about computers, computer-related equipment such as terminals, program-development procedures and data processing. You ought to know basically how computers and computer-related devices operate and be familiar with some of the "buzz words" of computer technology. You will acquire this knowledge partly as you actually use the computer to develop and execute FORTRAN programs. You will also learn about computers and data processing particularly as applied to business applications in the parallel course, Management 237.

Course Outline and Weekly Objectives

The semester is partitioned into 12 weeks. (To allow for holidays and some harder assignments, not all "weeks" are exactly 7 days in length). As the semester progresses you should systematically achieve each of the objectives listed below. You may also use this outline as a checklist for the final examination.

Week 1 (January 15 - January 24):

Topic: Introduction to computers, SIMPLE, and the FORTRAN programming language.

Objectives: You should be able to

a. Use SIMPLE to edit and run a FORTRAN program

b. Properly space and punctuate FORTRAN statements

c. Understand the rudiments of the READ, assignment, GOTO, IF, and PRINT statements

(This week is often the hardest because everything is new.)

Week 2 (January 25 - January 31):

Topic: FORTRAN arithmetic expressions, assignment statements, and computer arithmetic

Objectives: You should be able to

a. Recognize FORTRAN constants and variable names and be able to classify them as either INTEGER or REAL type

b. Evaluate FORTRAN arithmetic expressions

c. Write algebraic expressions in FORTRAN

d. Explain how the assignment statement works

(If people could add as fast as computers, computers would probably be out of business.)

Week 3 (February 1 - February 7):

Topic: Input/Output

Objectives: You should be able to

a. Understand the difference between formatted and unformatted input/output

b. Explain the operation of the format codes I, F, and X for both input and output

c. Determine the carriage control specification of a FORMAT statement

d. Use literals to appropriately label computer output

(This stuff is difficult for some people to master, but you gotta tell the computer where to put things and where to get things.)

Week 4 (February 8 - February 14):

Topic: Flow of control

Objectives: You should be able to

a. Convert a flowchart to FORTRAN code

b. Write relational expressions in FORTRAN

c. Use logical IF statements to selectively execute segments of code and to control loops

d. Explain the use of header and trailer cards

(Computers aren't very smart, but if you ask the right question they put on a good show.)

Week 5 (February 15 - February 21):

Topic: DO-loops

Objectives: You should be able to

a. Show the equivalence between DO-loops and counting loops constructed with IF and assignment statements

b. Make appropriate use of the DO-loop

c. Define the terminology associated with DO-loops

d. State the simple rules on the use of DO-loops

(Although you may hate repetitious work, computers love to do things over and over.)

Week 6 (February 22 - February 28):

Topic: More on DO-loops

Objectives: You should be able to

a. Explain the difference between normal and abnormal exits from DO-loops

b. Know when the CONTINUE statement is necessary

c. Know how to properly nest DO-loops

(Computers love to do things over and over so much, they often repetitively repeat within repetitions.)

Week 7 (March 1 - March 7):

Topic: More on input/output

Objectives: You should be able to

a. Explain for simple situations what happens when there are fewer or more variables in the input/output list than format codes.

b. Explain how to use slashes for multiple records

c. Use repetition factors to simplify format specifications

(To make the results look fancy, you need to know some tricks of the trade.)

Week 8 (March 8 - March 14):

Topic: Arrays

Objectives: You should be able to

a. Understand the difference between arrays and simple variables

b. Know how to declare and use one-dimensional arrays

Week 9 (March 15 - March 21):

Topic: More on arrays

Objectives: You should be able to

a. Know how to declare and use multidimensional arrays

b. Give the storage order for multidimensional arrays

c. Make use of array input/output and implied DO-loops

Week 10 (March 22 - April 4):

Topic: Subprograms

Objectives: You should be able to

a. Cite basic reasons for use of subroutines

b. Write and make use of simple function subprograms

(Break it up, chop it down, divide and conquer.)

Week 11 (April 5 - April 18):

Topic: More on Subprograms

Objectives: You should be able to

a. Explain how information is passed to and from subprograms

b. Write and make use of subprograms

c. Explain the similarities and differences between subprograms and functions.

Week 12 (April 19 - May 2):

Topic: Important algorithms and implementation considerations

Objectives: You should be able to

a. Understand how to write routines to sort, merge, search, and update files

b. Recognize time and space efficiency considerations

(By this time, you ought to be able to have the computer do something useful - like figure your income tax.)

If you meet all these objectives, you will be saturated with knowledge about FORTRAN. Relax and treat yourself to an ice-cream soda.

Scheduled Events

Since there are no lectures, the class wil not meet as a group except for the following events.

1. First scheduled class meeting during first week of classes.

   Course mechanics, account numbers, introduction to SIMPLE. Meet in Course Headquarters, Ferguson 17.

2. First scheduled class meeting during second week of classes.

   Answers questions on course mechanics, SIMPLE, and Fortran. Meet in Course Headquarters, Ferguson 17.

3. Film: "The Ultimate Machine"

   January 30 - February 1 (Tues-Thurs.) 9:30 a.m., 11:30 a.m., or 2:30 p.m.

   Attend one of nine showings in Course Headquarters, Ferguson 17.

4. Optional review for mid-semester examination.

   February 14-16 (Wed.-Fri.) 9:30 a.m., 11:30 a.m., or 2:30 p.m.

   Attend any number of sessions you wish (including none) in Course Headquarters.

5. Mid-semester examination.

   February 19-20 (Mon.-Tues.) 9:30 a.m., 11:30 a.m., 12:30 p.m., 2:30 p.m., and 3:30 p.m.

   You must sign up beforehand to reserve space for yourself at one session. A sign-up sheet will be provided in Course Headquarters approximately one week in advance for this purpose.

6. Tour of Lincoln Computing Facility.

   March 19-22 (Mon.-Thur.) 9:30 a.m., 2:30 p.m., 3:30 p.m., 4:30 p.m.

   March 23 (Fri.) 9:30 a.m., 2:30 p.m.

   The meeting place for all tours is Nebraska Hall, Room 233 - North entrance, second floor. In order to limit the group size to a reasonable number you will be asked to sign up for these tours in advance. A sign-up sheet will be available at Course Headquarters approximately one week before the first tour. Bring a friend.

7. Course evaluation and review for final examination.

   May 1-3 (Tues.-Thur.) 9:30 a.m., 11:30 a.m., and 2:30 p.m.

   Attend at least one session in Course Headquarters.

8. Final Examination.

May 7-11 (Mon.-Fri.)	10:00 a.m.-12:00 Noon
May 9,10 (Wed.-Thur.)	1:00 p.m.-3:00 p.m.
May 7-9 (Mon.-Wed.)	3:30 p.m.-5:30 p.m.

You must sign up beforehand to reserve space for yourself at one session. A sign-up sheet will be provided in Course Headquarters approximately one week in advance for this purpose.

COURSE MECHANICS

How to Successfully Complete the Course

There are twelve assignments, one for each of the weeks listed above, and two examinations, a mid-semester exam and a final exam.

For each assignment, there are four parts:

1. Understand the assigned reading material.
2. Work the assigned exercises.
3. Successfully write and run the assigned program(s).
4. Attend an oral quiz conducted by an instructor or teaching assistant.

The assignments can be picked up at course headquarters.

In order to guarantee that you will not fall behind in your work, there will be a hard and fast cutoff date and time for every programming assignment. It is easy to check this later during your oral quiz since the computer prints the date and time your program was run right on the listing of the results. No program run after your cutoff date and time will be accepted; however, partial credit will be given for incomplete programs run before that time.

Since there are many more students than computer terminals, a system of cutoff times staggered through the first three days of the week has been devised to reduce the congestion and bloodshed sure to occur if 400 students tried to use the thirty or so available terminals at the same time. During each of the twelve weeks that you have a programming assignment due, your cutoff day of the week and cutoff time during that day are determined by the following table:

First letter of last name	Cutoff Day	Cutoff time
A, B	Monday	12:00 noon
C, D	Monday	6:00 p.m.
E,F, G	Monday	12:00 midnight
H,I, J	Tuesday	12:00 noon
K, L	Tuesday	6:00 p.m.
M, N	Tuesday	12:00 midnight
O,P,Q, R	Wednesday	12:00 noon
S	Wednesday	6:00 p.m.
T - Z	Wednesday	12:00 midnight

You are responsible for completing each assignment in a timely manner. If you wish to do the assignments in advance, you may. In lieu of lectures, there is plenty of opportunity to get individual help, but you must take the initiative. If you fall behind, you will likely receive one or two warnings, but no one will stand over you and hold your nose to the grindstone.

We suggest that you plan your time carefully and avoid procrastination. Because of the number of people sharing the terminals, you have no guarantee that one will be free just when you decide to work on your program. Since most of the assignments are one week in length, you might plan your week as follows:

Day 1:

Read the assignment for the week.

Begin to read the assigned sections in the text (Read carefully; this is your main resource for learning new material.)

Day 2:

Finish the assigned reading material in the book. (If you have any questions or don't understand something, ask an instructor or teaching assistant.)

Begin to do the assigned exercises.

Day 3:

Finish the exercises. (If you have trouble, get some help.)

Begin to work on the programming assignment. (Don't put this off. Write what you want to do on paper before you sit down at a terminal. If you need help getting started, talk to an instructor or a TA.)

Day 4:

Work on your programming assignment. (If you are behind, now is the time to do something about it.)

Day 5:

Work on your programming assignment. (If you are having trouble, get help.)

Day 6:

Complete your programming assignment. (If you spent more time at the terminal than in preparation, you were probably not working as efficiently as you could be.)

Make an appointment for your oral quiz.

Day 7:

Watch your cutoff time pass with equanimity.

To make an appointment for your oral quiz, you should sign up on the sheets provided for this purpose at course headquarters. You may take your quiz any time before your cutoff time, but at the latest you must take it by 5:00 p.m. of the second day after your cutoff time. (This is done to prevent bunching up on Friday afternoon.) Whenever you take your oral quiz, remember that your programming assignment must be completed before your cutoff time.

Grading

Grades are a necessary evil. Here is how we plan to grade your achievement in this course:

	Points
Mid-semester examination	20
Final examination	40
Oral quizzes (including programs and exercises)	112
Optional program assignments	28
TOTAL	200

Letter grades will be assigned according to the following scheme:

Grade	Points
A	180 and above
B+	170 "
B	160 "
C+	150 "
C	140 "
D+	130 "
D	120 "
F	119 and below

Each examination will consist of questions similar to the exercises. The final examination will be a two-hour written test. Both examinations will be open book and of the same form.

The programming assignments will be graded by an instructor or a teaching assistant during the weekly oral quizzes. Each assignment will be worth 8 points except the last two worth 16 points each. The basis for the grade will be the printed output (including a listing of the program and of the program output) bearing your logon number and the date, and the answers you give to the questions intended to prove your understanding

of the material. You will be responsible for understanding the reading material assigned up to that date, the exercises, and how your program works.

The exact point breakdown of your oral quiz follows:

	Points
Properly working program	4
Appropriate comments on your listing, meaningful variable names, and clear program organization	1
Assigned exercises	1
Answers to questions over the material	2
TOTAL	8

The point values on the last two assignments will be doubled.

If the instructor or teaching assistant is convinced that you simply copied someone else's program without thoroughly understanding it, you will get zero for the entire assignment. Consider yourself warned that it is generally harder to understand someone else's program than to write your own!

The answers to each assignment will be given out on Thursday morning of the week the assignment is due. Therefore if you do not hand your assignment in on time, you will receive a zero. If you cannot complete the assignment because you are ill most of that week, the assignment will be waived on presentation of a note signed by your doctor or from the Student Health Service. If your motorcycle breaks down and you live 50 miles away, or you have to attend your uncle's wedding in Panama, we will accept a note signed by your academic advisor. Your final grade will be prorated for excused assignments.

On most weeks a more difficult optional programming assignment is given. If you solve this assignment successfully and can explain what you did, you will get additional points (usually two). Notice that the only way to get an A for the course is to do some of this extra credit work. A note of caution, however. If you do the extra credit work, the person giving you your oral quiz will expect you to demonstrate an excellent grasp of the material and understanding of your program. An incorrect program or gaps in your understanding of the material could cause you to lose more than you would have gained. This is not meant to discourage you from trying for an A but to inform you of our intention of giving top grades only for good work.

RESOURCES

The University, and specifically the Department of Computer Science, provides you with the resources necessary to learn how to program. It is up to you, however, to take advantage of these resources to master the material.

The following paragraphs describe the resources available to you roughly in the order that you should use them.

The Text

PROGRAMMING WITH FORTRAN
by Seymour Lipschutz and Arthur Poe
Schaum's Outline Series 1978 $5.65

This book was selected by the faculty from among thirty or so considered on the grounds of clarity, brevity, price, and examples and exercises suitable for a lectureless course. Not everyone, however, has the same taste in textbooks. If you wish to look at other books, four others are:

UNDERSTANDING FORTRAN
by Michel Bouillot
West Publishing Co. 1978 $11.50

FORTRAN
by Samuel Marateck
Academic Press 1977 $10.00

BASIC FORTRAN IV WITH WATFOR AND WATFIV
by C. Bauer and A. Pelluso
Addison-Wesley 1974 $9.95

FORTRAN FOR BUSINESS PEOPLE
by Didday, Page, Hagen, Wayne
West Publishing Co. 1978 $11.00

These books are available at Course Headquarters (unless other students remove them.)

Even the best programming texts make slow reading - give yourself plenty of time and don't attempt to read the assigned sections as if they were science fiction. One of the skills you must master for computer programming is meticulous, detailed thinking. To understand the examples, you will have to follow them almost letter by letter. You will find that such attention to detail is essential to write correct programs, and will save you time in the long run.

The Computer

You can't learn how to program just by reading or watching - it is a "hands-on" skill. You will practice it at one of the many computer terminals

available. After you have read the assignment and thought through and written down one or more possible solutions to a problem, try it out on the computer.

When you are working at a terminal, you don't have to worry that by doing something wrong you will cause a catastrophe. Feel free to try new things. At worst, you will get some answer from the computer that does not make sense to you. Remember, however, that you are being charged at about $2.50 per hour when you are connected to the computer, hence if you do not want to exhaust your computer allocation it is best to think ahead of time about what are you going to do. This may also help you avoid the mistakes that are usually introduced when you do things on the spur of the moment.

The computer system you will use in this course is an IBM/370 Model 158 computer located in the State Capitol. Terminals are connected to the computer through telephone lines. In order to use this computer, you must dial the computer's number on the telephone located near each terminal. Other terminals on Campus are connected directly (without a telephone) to another IBM computer at the Lincoln Computer Facility (LCF) in Nebraska Hall or to a Control Data computer in Minneapolis. You will not use these computers, but we have scheduled a tour of the Nebraska Hall computer facility (machine room) to give you a first hand look.

Course Headquarters (CH)

Ferguson Room 17 serves as course headquarters. It is a place where you can get help and explanations, prepare your work before you log-on to a terminal, study your results, and discuss problems (or anything else) with other students in the course. As mentioned, we also plan to keep reference materials on hand and, if our budget holds up, doughnuts once in a blue moon. Please consult the CH bulletin board frequently for course-related announcements.

Instructors

Course instructors are computer science faculty who have ultimate responsibility for the course. They have regular office hours which are posted on the CH bulletin board. You will also find them from time to time in the CH or near the terminals. You will have about one third of your oral quizzes with one of the instructors. You may, of course, also see them

during office hours or, if you have a longer topic to discuss with them, make an appointment. Since there are no lectures, we expect that you will learn much of the information you can't find in the text from discussions with the instructors and teaching assistants. The instructors are also responsible for your final grade for the course.

Teaching Assistants

Teaching assistants are computer science graduate students and undergraduate majors with a good academic record. They will answer your questions about FORTRAN, SIMPLE, exercises, assignments, and the mechanics of the course itself. They will also conduct many of the weekly oral quizzes.

We plan to have teaching assistants stationed at course headquarters 40 hours a week according to the schedule below. A detailed schedule showing who is available when will be posted in the CH. The telephone number there is 472-3286.

Teaching assistants have offices elsewhere in Ferguson Hall, but since they have other duties as well, and must also study for their own courses, please approach them only with the most burning problems when they are not on duty.

	M	T	W	T	F	S	S
08:30							
	Yes	Yes	Yes	Yes	Yes		
11:30							
14:00	Yes	Yes	Yes	Yes			Yes
17:00							
20:00	Yes	Yes	Yes	Yes			Yes
22:00							

Debug Consultants

Debug consultants have desks and posted hours in the terminal rooms in Ferguson Hall, Nebraska Hall, and Miller Hall. As their red badge shows, they work for the University of Nebraska Computing Network (UNCN) which is responsible for the operations and maintenance of the computer systems available to the students and staff. Debug consultants usually have had three or four computer courses and have passed a test administered by UNCN.

Consultants may help you with problems not directly related to your assignment - if, for instance, you want to try to write a program on your own or experiment with some aspect of the system or language not covered in the course. Consultants can also help you log on or off the terminals, find material in reference manuals and programmers' guides, and interpret strange error messages from the computer. For help with assignments, however, please turn instead to the teaching assistants and instructors.

I/O Clerks

I/O clerks also work for UNCN; many of them are students. They distribute the printer output in Ferguson Hall and in Nebraska Hall. Since the printers are connected to the Nebraska Hall computers which you will not use in this course, you will have little to do with the I/O clerks. They are, however, also responsible for reporting equipment malfunctions to the Nebraska Hall operations crew which, in turn, notifies the manufacturers with whom the University has maintenance contracts. If you notice problems with your terminal, such as sticky keys, no paper, or worse, please report them, preferably on a slip of paper, to the I/O clerk on duty. You may also report them by telephone to the receptionist in the Nebraska Hall Computing Facility, at 472-3701.

Other Students

Don't overlook the fact that there are probably other students in the class who may know the answers to some of your problems. Although you should do the early assignments on your own to master the necessary details, you will be able to do later assignments in cooperation with other students. In the meantime, discuss your problems with them, and look at their solution. The oral quizzes, however, are intended to ascertain that each student understands every detail of the program he presents and of the assigned exercises.

HOW TO USE THE TERMINALS

Terminal Location and Hours

The CMS system (that is how it is usually referred to) is normally available 24 hours a day, seven days a week. You can dial up a connection from any of about 60 terminals in Lincoln, and from 4 or 5 in Omaha. If you have a terminal of your own ($600-$1,500), and if you don't mind tying up your telephone, you can even dial up from home! Public terminals in Lincoln are available at the following locations and hours (both subject to change - terminals, nowadays, are quite portable):

location	hours	terminals
Ferguson Room 20	24 hours a day 7 days a week	9 Decwriters 4 ADDS display terminals
Ferguson Room 06	Same	2 IBM 2741 Selectrics 1 Tektronix terminal
(On Sundays after 11:00 p.m. and on football Saturdays only the West door of the building is open)		
Miller Hall Room 104 (East Campus near Student Center)	8:00 a.m.-11:00 p.m. (Monday - Friday) 8:00 a.m.-5:00 p.m. (Saturday) 1:00 p.m.-5:00 p.m. (Sunday)	10 Decwriters 1 Tektronix terminal
Nebraska Hall Room 233	24 hours a day 7 days a week	1 Decwriter 1 IBM 2741 terminal 1 ADDS display terminal 1 Tektronix terminal
(Use North entrance to Nebraska Hall)		
CBA Room 143	8:00 a.m.-5:00 p.m. (Monday - Saturday)	4 Decwriters 1 IBM 2741 terminal

Sometimes (not very often) the computer system is down for repairs. You can check whether it is up (working)by calling 1-9575. If the ring is followed by a high-pitched tone, everything is in order. If not, call 472-3701 for information about the status of the CMS computer. It may also happen that all lines to the computer are busy; if you get the busy signal, hang up and try again later.

Logging On and Off

The telephone handsets near each terminal are already wired to the terminal. Therefore all you have to do to log on (make connection with

the computer) is to dial the number of the computer. When the computer answers (with the high-pitched tone), pull up the white button on the handset, put the telephone receiver on the table, and type the information necessary to tell the computer that you are authorized to use it. The details of this procedure differ slightly from terminal to terminal. The following paragraphs explain how to log on with the most common terminals. If you have trouble, ask a nearby "expert", a consultant, or a teaching assistant.

Instructions for logging on the different types of terminals

The telephone number of the computer is 1-9575.

Decwriter terminals

The on/off switch is near the top left corner of the keyboard. Among the cluster of eight keys at the left, only the HDX/FDX and the 300 buttons should be depressed. Note that on DECWRITER terminals the Carriage Return key (on the right) is marked RETURN. You must press it after you type in each line (including the logon number and the password).

ADDS display terminals

The on/off switch is the black button near the top right of the keyboard. The Carriage Return Key is marked NEWLINE.

All terminals

Whichever terminal you use, after turning the terminal on, dial 1-9575, pull up the white button when you hear the tone, and wait for the message "VM/370 ONLINE". Press Carriage Return (CR), then type "logon" followed by your logon number and another CR; for example

logon csc4547 CR (CSC4547 is an example of a logon number)

If you make a mistake, the computer will say: RESTART, and you have to type logon csc4547 again. If your logon number is accepted by the computer, it will say:

ENTER PASSWORD:

The blacked out characters ("mask") are there to keep other people from seeing your password.

Type your 8 digit password.

The computer will print out:

LOGON AT 10:44:06 CST SATURDAY 08/09/78 (with the current date and time, of course)
CMS 5.04C

When the dct is typed, the computer is checking to see if you are still there, so enter a carriage return (null line). The computer will now print out:

PLEASE TYPE YOUR SIMPLE NUMBER.

Now type in the last four digits of your social security number, for example

3897 CR (this is your SIMPLE number).

Next, the computer will type

YOU MAY NOW EDIT filename (contains n lines)

where filename represents the name of the file you have just accessed. You are now ready to do some productive work.

When you finish, logoff, turn the terminal off, and hang up. If you hang up without logging off you will loose any changes you made in your files since you last logged on (i.e., during the current terminal session), and you will be charged for about 15 minutes of time before the computer notices that you are no longer connected.

Etiquette

The terminals are used by hundreds of students and university personnel, therefore some goodwill and cooperation is necessary to let everyone get on with the work at hand.

If there are people waiting to use the terminals, limit your terminal session to at most half-an-hour. As you will soon find out, some times of the day are much busier than others. Mealtimes and weekends are least busy; therefore if you are desperate, skip a meal or a weekend quilting bee.

Space around the terminals is limited. If you carry your entire library with you, stack some of it near your feet. Please do not leave any printouts near the terminal. Discard whatever you don't need in the receptacles provided for the purpose. (Recycling the paper helps keep the University of Nebraska solvent.) Before you leave, advance the paper to leave a clean edge at the top - it is nice to sit down at a terminal cleared for action. For conversation, adjourn to Ferguson 17 or the corridor; it is hard enough to find mistakes without having to listen to people chattering. If you want to eat or drink, there is a vending machine room on the first floor of Ferguson Hall.

REPORT TERMINAL MALFUNCTIONS TO THE I/O CLERK.

Computer Accounts

Each student has his own computer account. The Department of Computer Science puts money into these accounts to enable students to perform the work necessary for the course. You will normally have between $5.- and $10.- to carry out each assignment.

You alone are responsible for the security of your account. Therefore take precautions to keep your account number and passwords to yourself. If someone else acquires these numbers and depletes your account, you are in trouble.

At the beginning of the semester, you will be given two punched IBM cards with your account number, such as CSC3658, and your 8 digit password, such as 38162497, written on it. You will put your name and complete social security number on one and give it back to the instructor. You will keep the other one. It contains your personal logon number that you will use for the rest of the semester. You will need to know these numbers every time you want to use the computer.

When you logon, you must first enter your account number, as explained earlier. Then you will type your password in the blacked out area that the computer provides for you. Next, you will be asked to enter your SIMPLE number, which consists of the last four digits of your social security number. The very first time you logon with your account number you will also be asked your name, first name and then last name. SIMPLE will remember your name because it doesn't want to think of you as just a number.

No one else can access your files unless they know <u>both</u> your social security number and your logon number. <u>Keep these to yourself</u>! Once you have given them out to somebody else, there is no way to get them back: it is like giving out the key to your safe deposit box.

Please note that the instructor <u>can</u> access your files. Furthermore, the computer keeps some statistical information on your computer usage. At the end of each session, the computer will inform you how long the session lasted, how much it cost, and how much you have left in your account.

If you have problems with your account number, see an instructor or a teaching assistant.

Introduction

SIMPLE lets you communicate with the computer. With SIMPLE you can create programs, alter them, and submit them for execution. You can also write, read, and edit ordinary English text and send and receive messages. Just sit down at a terminal (a typewriter-like device connected to the computer) and issue commands to tell the computer what you want done. Since the computer does not understand English, however, you must issue your commands in a language it understands. That's where SIMPLE comes in.

SIMPLE is a command language (as opposed to a programming language, like FORTRAN) for beginners. It has only a few commands. These commands are easy to learn and they provide you with capabilities sufficient for most ordinary applications. After you gain a little experience, you may wish to learn some other command language such as CMS or NUROS. These provide additional features that can enhance your ability to communicate effectively with the computer and with computer users at other terminals.

Some Necessary Explanations

Files. The SIMPLE command language lets you read and alter files. A file is a named group of lines of text. The text may be FORTRAN code or ordinary English. Some files are read-only; others are read/write. Read-only files can be read but not altered (they are like a library book); read/write files can be read and altered (they are like your own notes). When you first sit down and begin to use SIMPLE, a file (possibly empty) is automatically provided for you.

Behavior. SIMPLE is either attentive or active. When it types >>> and waits it is attentive. You are expected to type a command and enter it by pressing the carriage return. SIMPLE then becomes active and either obeys the command or types an error message to explain why it cannot obey the command. Afterwards, it again types >>> and is attentive.

Typo's. If you make a typing error, strike the "backspace" key if you use a Decwriter terminal. If you use an ADDS terminal, you must simultaneously press the CONTROL key on the left and the h key! This effectively erases the character just entered and allows you to continue from that point. You may erase several characters by successive backspaces.

Commands. SIMPLE has 15 commands:

APPEND	GET	OPEN
BACK	HELP	PUT
CHANGE	INSERT	QUIT
DELETE	LOCATE	RUN
FORWARD	NAMES	TYPE

Note that the first letter uniquely identifies a command. Accordingly, you may also specify a SIMPLE command by the first letter. To make things easier in the beginning, your instructor may limit your use of SIMPLE to only a few of these commands, then, after a week or two of experience, you will likely be allowed to use all of them.

Command Format. SIMPLE commands consist of a command keyword (listed above) often followed by an appropriate argument (a number or a sequence of characters that makes the command more specific). The argument is optional. If absent, a reasonable value ("default") is assumed when necessary. In the explanations below, command keywords appear in upper case, and arguments appear in lower case.

Current Line. SIMPLE keeps track of where you are so it will know what to change when you make a request. Commands are issued relative to a current line. Initially, the current line is at the top of the file. Thereafter, the position of the current line depends on what you do. The current line is often typed out after a command is executed.

Carriage Return. Each command or line you wish to enter in the computer must be followed by a Carriage Return. If you don't press Carriage Return, the information you just typed is not transmitted to the computer, and it just waits for you. On all terminals, the Carriage Return (CR) is near the right edge of the keyboard. On Decwriters it is labeled RETURN. On ADDS terminals it is marked NEWLINE. On IBM 2741's, it is CR.

Common Typing Errors.

1) Using lower case l instead of the numeral 1 (one);
2) Using the letter O instead of the numeral 0 (zero) and vice versa;
3) Using the letter I when the numeral 1 is meant and vice versa.

The numeric keys are in the top row of the keyboard.

BEGIN. Sometimes (rarely), noise on the telephone line or other problems cause the computer to take you out of the SIMPLE system. Then the computer will type CMS or CP to let you know that there is a problem. If this happens, type begin to get back to where you were.

Some Elementary Commands

INSERT. Insert text after the current line. After issuing this command, you may type in as many lines of text as you wish, each followed by a carriage return. When finished, enter an additional carriage return (i.e., a null line) to signal that you are through.

DELETE n. Delete n consecutive lines beginning with the current line. If n is omitted, one line is deleted.

FORWARD n. Move forward n lines from the current line. If n is omitted, move forward one line.

BACK n. Move back n lines from the current line. If n is omitted, move back one line.

TYPE n. Type n consecutive lines beginning with the current line. If n is omitted, only the current line is typed.

QUIT. Terminate the session.

HELP. List the commands available.

HELP command. Give an explanation about the command specified. For instance,

HELP INSERT

would cause the computer to print out an explanation of the INSERT command.

Example 1

To illustrate these elementary commands, let's look at a complete terminal session. As stated above, a file is automatically provided for you. Suppose this file is called RHYME and initially contains the following nursery rhyme.

HEY DIDDLE DIDDLE
THE CAT AND THE VIOLIN
THE COW JUMPED OVER THE MOON
THE L
ITTLE DOG LAUGHED TO SEE SUCH SPORT
AND THE DISH RAN AWAY WITH THE SPOON

As you can see, there are some errors. "VIOLIN" should be "FIDDLE", and Lines 4 and 5 should be

THE LITTLE DOG LAUGHED
TO SEE SUCH SPORT

Let's see how we can fix it using SIMPLE. First, you turn the terminal on and then make connections with the computer by dialing 1-9575. When the computer answers (with a high-pitched tone), pull up the white button on the hand set. The computer then types the first line shown in the dialogue below.

Dialogue	Explanation
	The text typed by the computer is capitalized, and the text typed by the user (that's you) is not capitalized and is preceded by a dot. After each line you enter you must press Carriage Return (not shown.
VM/370 ONLINE !	Log-on sequence for decwriter terminals (the logon section of the Course Guide contains additional details). Note that after the exclamation mark is printed, you must enter a carriage return.
.log un17503 ENTER PASSWORD: .■■■■■■■■■	
DASD 190 LINKED R/O; R/W BY MAINT; R/O BY 041 USERS LOGON AT 10:42:45 CST TUESDAY 08/22/78 CMS 5.04C	
.	Enter a carriage return when the dot appears.
A (191) R/O PLEASE TYPE IN YOUR SIMPLE NUMBER. .■■■■■■	
YOU MAY NOW EDIT RHYME (CONTAINS 6 LINES)	You are in the file called RHYME
$TOP OF RHYME	This is the beginning of the file RHYME.
>>>	>>> prompts you to enter a command.
.type 99	Types out the file. (Types 99 lines or to
$TOP OF RHYME	the end of the file - whichever comes first.)
HEY DIDDLE DIDDLE	Note the beginning and end of file markers -
THE CAT AND THE VIOLIN	"$TOP OF RHYME" and "$END OF RHYME". These
THE COW JUMPED OVER THE MOON	aren't actually part of the file but mark
THE L	the beginning and end respectively. You can-
ITTLE DOG LAUGHED TO SEE SUCH SPORT	not delete or change these markers.
AND THE DISH RAN AWAY WITH THE SPOON	
$END OF RHYME	
>>>	
.forward 2	Moves forward 2 lines and the new current
THE CAT AND THE VIOLIN	line is typed out.
>>>	
.delete	Deletes the current line.
1 LINES DELETED	
HEY DIDDLE DIDDLE	Types out the new current line.
>>>	
.type 2	Types out 2 lines (to illustrate that the old
HEY DIDDLE DIDDLE	unwanted line between them is gone).
THE COW JUMPED OVER THE MOON	
>>>	
.insert	Indicates that lines are to be inserted.
START TYPING	Prompts the user to start typing.
.the cat and the fiddle	
.	An empty line terminates insertion.

Dialogue	Explanation
>>>	
.type 3	Types out 3 lines (to illustrate that the new line is inserted).
HEY DIDDLE DIDDLE	
THE CAT AND THE FIDDLE	
THE COW JUMPED OVER THE MOON	
>>>	
.forward 3	
THE L	
>>>	
.delete 2	Requests 2 lines to be deleted.
TO DELETE 2 LINES, TYPE Y, ELSE TYPE N	Protects you so you don't inadvertently delete several lines of text.
.y	
2 LINES DELETED	
THE COW JUMPED OVER THE MOON	Types out the new current line.
>>>	
.type 2	Types out 2 lines (to illustrate that the old unwanted lines in between are deleted).
THE COW JUMPED OVER THE MOON	
AND THE DISH RAN AWAY WITH THE SPOON	
>>>	
.insert	
START TYPING	
.the little dog laughed	
.to see such sport	
.	An empty line terminates insertion.
>>>	
.back 99	Moves back 99 lines or to the top of the file - whichever comes first.
$TOP OF RHYME	
>>>	
.type 99	Types out the file.
$TOP OF RHYME	
HEY DIDDLE DIDDLE	
THE CAT AND THE FIDDLE	
THE COW JUMPED OVER THE MOON	
THE LITTLE DOG LAUGHED	
TO SEE SUCH SPORT	
AND THE DISH RAN AWAY WITH THE SPOON	
$END OF RHYME	
>>>	
.quit	
CHARGE $ 1.19	
BALANCE $ 38.66	
CONNECT= 00:11:00 VIRTCPU= 000:16.27 TOTCPU= 000:29.30	
LOGOFF AT 16:14:17 CST MONDAY 08/21/78	

Executing a program

If you want to execute a program, you need the RUN command. For this command to work properly, the text in the file you're working with must of course be a FORTRAN program. When you issue the RUN command, the program executes and then types out the error diagnostics (if any), the results, and a listing annotated with line numbers and account information.

Example 2

Suppose you wish to create and run the following FORTRAN program.

```
C CALCULATE SUM AND PRODUCT
      I = 24
      J = 3
      LSUM = I+J
      LPROD = I*J
      PRINT, I,J,LSUM,LPROD
      STOP
      END
```

Suppose further that an empty file called PGM is provided for you.

After logging on, you continue as shown below.

Dialogue	Explanation

```
YOU MAY NOW EDIT PGM ( CONTAINS 0 LINES )
$TOP OF PGM
>>>
.insert
START TYPING
.c calculate sum and product
.       i = 24
.       j = 3
.       lsum = i+j
.       lprod = i*j
.       print, i,j,lsum,lprod
.       stop
.       end
.
>>>
.run
PGM SUBMITTE FOR EXECUTIO - STAND BY FOR RESULTS
$RESULTS

         $JOB            TIME=10
         C CALCULATE SUM AND PRODUCT
     1           I = 24
     2           J = 3
     3           LSUM = I+J
     4           LPROD = I*J
     5           PRINT, I,J,LSUM,LPROD
     6           STOP
     7           END
         $ENTRY
          24            3           27           72
CORE USAGE      OBJECT CODE=      320 BYTES,ARRAY AREA=        0
DIAGNOSTICS       NUMBER OF ERRORS=        0, NUMBER OF WARNINGS=        0
COMPILE TIME=     0.02 SEC,EXECUTION TIME=      0.00 SEC
         $STOP


>>>
.quit
```

You are in the empty file called PGM.

You must type in your program.

An empty line terminates insertion.

Request for compilation and execution.

Listing of your program.

Program output.

Remember, you must use "quit" to logoff.

Example 3

Suppose you now wish to alter this program so that instead of assigning I the value 24 and J the value 3 you read values into I and J. Since the new program will read values for I and J, these input data values must be supplied. When using SIMPLE, you place input data at the end of the program, separated from the program by the single line of text

$DATA

positioned in columns 1-5.

After making the appropriate changes, the file PGM should look like this:

```
C CALCULATE SUM AND PRODUCT
      READ, I,J
      LSUM = I+J
      LPROD = I*J
      PRINT, I,J,LSUM,LPROD
      STOP
      END
$DATA
  24  3
```

Note that the second and third line have to be replaced by the READ statement and that the input data has to be added at the end. After logging on, the following dialogue will accomplish the necessary changes.

Dialogue	Explanation
YOU MAY NOW EDIT PGM (CONTAINS 8 LINES)	The computer stored the eight lines you created last time in the file called PGM, which is automatically made available to you when you logon.
$TOP OF PGM	
>>>	
.type 99	
$TOP OF PGM	
C CALCULATE SUM AND PRODUCT	
I = 24	
J = 3	
LSUM = I+J	
LPROD = I*J	
PRINT, I,J,LSUM,LPROD	
STOP	
END	
$END OF PGM	
>>>	
.forward 2	Move forward to the lines to be deleted.
I = 24	
>>>	
.delete 2	Delete 2 lines.
TO DELETE 2 LINES, TYPE Y, ELSE TYPE N	
.y	
2 LINES DELETED	
C CALCULATE SUM AND PRODUCT	New current line.
>>>	
.insert	Insert READ statement.
START TYPING	
. read, i,J	
.	
>>>	
.forward 99	Move forward to the end of the program.
$END OF PGM	
>>>	
.insert	Insert the data.
START TYPING	
.$data	
. 24 3	
.	
>>>	
.back 99	Move back to the top of the file.
$TOP OF PGM	
>>>	
.type 99	As a check, type out the new program and data.
$TOP OF PGM	
C CALCULATE SUM AND PRODUCT	
READ, I,J	
LSUM = I+J	
LPROD = I*J	
PRINT, I,J,LSUM,LPROD	
STOP	
END	
$DATA	
24 3	
$END OF PGM	

```
>>>
.run
PGM SUBMITTE FOR EXECUTIO - STAND BY FOR RESULTS
$RESULTS

          $JOB              TIME=10
          C CALCULATE SUM AND PRODUCT
      1           READ, I,J
      2           LSUM = I+J
      3           LPROD = I*J
      4           PRINT, I,J,LSUM,LPROD
      5           STOP
      6           END
          $ENTRY
           24              3              27              72
CORE USAGE          OBJECT CODE=      328 BYTES,ARRAY AREA=         0
DIAGNOSTICS           NUMBER OF ERRORS=         0, NUMBER OF WARNINGS=         0
COMPILE TIME=      0.02 SEC,EXECUTION TIME=      0.00 SEC
          $STOP


>>>
.quit
```

Don't forget the "quit" command. Wait until the account information is displayed before you hang up, otherwise you may loose your file!

LOCATE and CHANGE

You can make any changes you need to with the commands presented so far. The LOCATE and CHANGE commands, however, can make your job easier. LOCATE lets you automatically find a line that contains a known sequence of characters so you don't have to search for the line with FORWARD, BACK, and TYPE. CHANGE lets you replace one sequence of characters by a different sequence of characters so you don't always have to use DELETE and INSERT to alter existing text.

LOCATE /x/. Locates the line that contains the sequence of characters x, enclosed in slashes. The search for x begins with the line following the current line and continues to the end of the file. For example.

LOCATE /LSUM/

issued when the current line is at the top of the file would locate the first occurrence of LSUM in the file and make the line containing LSUM the new current line.

LOCATE. Locates the line that contains the sequence of characters specified most recently in a LOCATE command. In other words, if you want to locate successive occurrences of the same sequence of characters, you don't have to type them again.

CHANGE. Change some sequence of characters in the current line to something else. When you issue a CHANGE command, SIMPLE asks you what you want to change and what you want to change it to and then alters the line accordingly. For example, suppose you wish to change LSUM to TOTAL in the following line of code.

```
PRINT, I,J,LSUM,LPROD
```

You would proceed as follows.

Dialogue	Explanation
`>>>`	
`.change`	Requests a change to be made.
`CHANGE WHAT?`	
`.lsum`	LSUM to be changed.
`TO WHAT?`	
`.total`	To be changed to TOTAL.
`PRINT, I,J,TOTAL,LPROD`	Types changed line.

CHANGE /x/y/. Change all occurrences of the sequence of characters x in the current line to the sequence of characters y. This is a short form of the CHANGE command where the character sequences are specified as part of the command. In the example above, the command

```
CHANGE /LSUM/TOTAL/
```

produces the same result.

CHANGE /x/y/ n. Change all occurrences of the sequence of characters x to the sequence of characters y in the next n lines starting with the current line. This lets you make the same change in several lines at once.

Example 4

As an illustration of the LOCATE and CHANGE commands, consider the program used in Example 5 to calculate the sum and product of 10 pairs of numbers. Suppose you wish to change the variable named N in this program to CTR.

You might be tempted to simply issue the command

```
CHANGE /N/CTR/ 99
```

but this doesn't yield the desired result as shown below.

Dialogue	Explanation

```
>>>
.type 99
$TOP OF PGM
C  CALC SUM AND PRODUCT OF 5 PAIRS OF NUMBERS
      N = 0
   20 READ, I,J
      LSUM = I+J
      LPROD = I*J
      PRINT, I,J,LSUM,LPROD
      N = N+1
      IF(N.LT.5) GOTO 20
      STOP
      END
$END OF PGM
>>>
.change /n/ctr/ 99
C  CALC SUM ACTRD PRODUCT OF 5 PAIRS OF CTRUMBERS
      CTR = 0
      PRICTRT, I,J,LSUM,LPROD
      CTR = CTR+1
      IF(CTR.LT.5) GOTO 20
      ECTRD
>>>
.type 99
$TOP OF PGM
C  CALC SUM ACTRD PRODUCT OF 5 PAIRS OF CTRUMBERS
      CTR = 0
   20 READ, I,J
      LSUM = I+J
      LPROD = I*J
      PRICTRT, I,J,LSUM,LPROD
      CTR = CTR+1
      IF(CTR.LT.5) GOTO 20
      STOP
      ECTRD
$END OF PGM
```

Type out the original program.

Change all occurrences of N to CTR. (Automatically prints each altered line.

Type out altered program.

As you can see, all of the N's were changed instead of just those that you wanted changed. At this point you would have to fix your mistake, and in this case, probably the best way to proceed is to change all the CTR's back to N's and start over. For the example, however, let's just start from the beginning and do it the right way.

Dialogue	Explanation

```
>>>
.type 99                                  Type original version of the program.
$TOP OF PGM
C  CALC SUM AND PRODUCT OF 5 PAIRS OF NUMBERS
      N = 0
   20 READ, I,J
      LSUM = I+J
      LPROD = I*J
      PRINT, I,J,LSUM,LPROD
      N = N+1
      IF(N.LT.5) GOTO 20
      STOP
      END
$END OF PGM
>>>
.locate /n/                               Locate the first line with an N.
C  CALC SUM AND PRODUCT OF 5 PAIRS OF NUMBERS   This line not to be changed.
>>>
.locate                                   Locate the next line containing N.
      N = 0
>>>
.change /n/ctr/                           Change N to CTR in this line.
      CTR = 0
>>>
.locate                                   Locate the next line containing N.
      PRINT, I,J,LSUM,LPROD               This line not to be changed.
>>>
.locate                                   Locate the next line containing N.
      N = N+1
>>>
.change /n/ctr/ 2                         Change the N's in this and the next line to CTR.
      CTR = CTR+1
      IF(CTR.LT.5) GOTO 20
>>>
.back 99
$TOP OF PGM
>>>
.type 99                                  Type altered program.
$TOP OF PGM
C  CALC SUM AND PRODUCT OF 5 PAIRS OF NUMBERS
      CTR = 0
   20 READ, I,J
      LSUM = I+J
      LPROD = I*J
      PRINT, I,J,LSUM,LPROD
      CTR = CTR+1
      IF(CTR.LT.5) GOTO 20
      STOP
      END
$END OF PGM
```

Slash Problem. The one problem with the LOCATE and CHANGE command is that you may wish to locate or change some sequence of characters that contains a slash. If you try to change a sequence of characters that contains a slash it won't work. The command

CHANGE /8/6/78/8/78/

is an attempt to change '8/6/78' to '8/78' but is ambiguous. The computer can't tell which slashes are part of the command and which are part of the text.

To solve this problem for both the LOCATE and the CHANGE command, SIMPLE allows you to substitute any character for the slash in the command. The computer can always tell which character stands for the slash because it is the first nonblank character following the command keyword.

To make the change indicated above, you could use a dollar sign for the slash and issue the command

CHANGE $8/6/78$/8/78$

Simple enough? - right, it's SIMPLE.

Working With More Than One File

So far you've only seen how SIMPLE works with one file, and this file has always been provided for you. SIMPLE lets you maintain several files and create some of your own. You can always find out what files are available by issuing the NAMES command. This command directs the computer to list the names of all your files whether you created them or whether they were provided for you.

To gain access to one of these files, you open it with the command

OPEN filename

where "filename" is the name of one of your files. This command moves you from the file you are working with to the file you specify. The file you just left is saved in exactly the form you left it, and you may open it again later. (Note, however, that if you hang up without logging off, or are accidentally disconnected, all file modifications made in the current terminal session are lost.)

If you wish to create a new file, choose a name that is not on your list of filenames and issue the OPEN command specifying the chosen name. (If you choose a name with more than 8 characters, the computer will use only the first 8.) If you wish to destroy an old file, simply open it and delete all the lines in it with the DELETE 999 command.

You can also transfer information from one file to another. The procedure, in essence, is to open the first file, copy the desired lines of text into a save area, open the second file, and copy the text in the save area into a specified place in the second file. The first file is left intact (the copy process is non-destructive).

The SIMPLE commands PUT, GET, and APPEND are provided to carry out this information transfer:

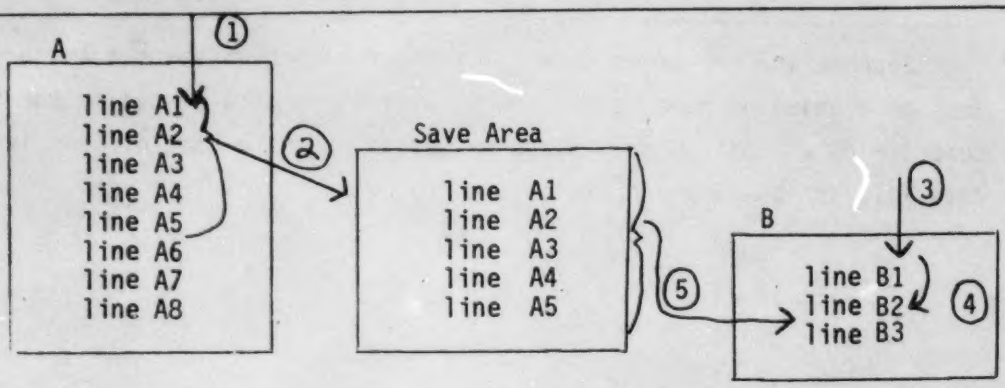
PUT n. Clear the save area and put n consecutive lines beginning with the current line into the save area. If n is omitted, the current line is put into the save area.

GET. Get all of the text in the save area and insert it after the current line.

APPEND n. Append n consecutive lines beginning with the current line to the text in the save area. If n is omitted, the current line is appended. This command is similar to PUT, but it does not clear the save area. It is useful if you are not sure ahead of time how many lines you want to transfer.

Figure 1 illustrates how the PUT and GET commands can be used to transfer the first five lines from file A into file B, between the second and third lines.

A common use of PUT and GET is allow an instructor to provide data for a program you write. An instructor can set up an input data file, give it a name, and place it on the system for your use. When you want to use the data, simply copy it into your program file at the bottom.



1. OPEN A
2. PUT 5
3. OPEN B
4. FORWARD 2
5. GET

Contents of files after transfer:

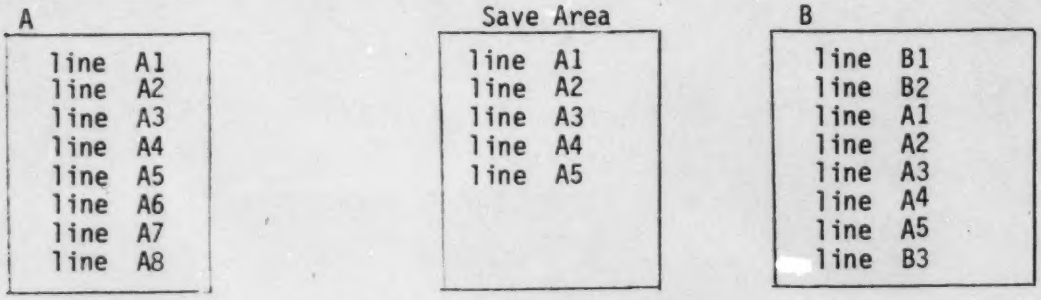


Figure 1. Transfer of Lines from File A to File B Using PUT and GET.

Example 5

Suppose you are asked to write a program to find the sum and product of 5 pairs of numbers and run it with input data placed in the the computer by an instructor. Suppose further that the name of this input data file is INDATA and contains the following.

```
$DATA
12 24
36 98
45 98
37 17
83 49
```

A FORTRAN program for solving this problem looks like this:

```
C  CALC SUM AND PRODUCT OF 5 PAIRS OF NUMBERS
      N = 0
   20 READ, I,J
      LSUM = I+J
      LPROD = I*J
      PRINT, I,J,LSUM,LPROD
      N = N+1
      IF(N.LT.5) GOTO 20
      STOP
      END
```

Assume you are initially given an empty file called PGM. After logging on, you can proceed as follows to write the code, get the data, and run the program.

Dialogue	Explanation

```
YOU MAY NOW EDIT PGM ( CONTAINS 0 LINES )
$TOP OF PGM
>>>
.names                                  Request the names of your files.
PGM       WRITE     CONTAINS 0    LINES
INDATA    READ      CONTAINS 6    LINES
>>>
.insert                                 Insert program.
START TYPING
.c  calc sum and product of 5 pairs of numbers
.       n = 0
.    20 read, i,j
.       lsum = i+j
.       lprod = i*j
.       print, i,j,lsum,lprod
.       n = n+1
.       if(n.lt.5) goto 20
.       stop
.       end
.
>>>
.open indata                            Open data file.
YOU MAY NOW READ INDATA ( CONTAINS 6 LINES )
$TOP OF INDATA
>>>
.put 99                                 Put the file in the save area.
6 LINES SAVED
>>>
.open pgm                               Go back to the program (in file PGM).
YOU MAY NOW EDIT PGM ( CONTAINS 10 LINES )
$TOP OF PGM
>>>
.forward 99
$END OF PGM                             Move to the end of the program where you
>>>                                     have to add the data.
.get                                    Get the data out of the save area.
6 LINES TRANSFER INTO PGM
>>>
```

```
.run
PGM SUBMITTE FOR EXECUTIO - STAND BY FOR RESULTS
$RESULTS

        $JOB              TIME=10
        C  CALC SUM AND PRODUCT OF 5 PAIRS OF NUMBERS
     1        N = 0
     2     20 READ, I,J
     3        LSUM = I+J
     4        LPROD = I*J
     5        PRINT, I,J,LSUM,LPROD
     6        N = N+1
     7        IF(N.LT.5) GOTO 20
     8        STOP
     9        END
        $ENTRY
         12              24              36             288
         36              98             134            3528
         45              98             143            4410
         37              17              54             629
         83              49             132            4067
CORE USAGE        OBJECT CODE=      416 BYTES,ARRAY AREA=        0
DIAGNOSTICS         NUMBER OF ERRORS=        0, NUMBER OF WARNINGS=        0
COMPILE TIME=     0.03 SEC,EXECUTION TIME=     0.01 SEC
        $STOP

>>>
.quit
```

If you hang up without the quit command, you will loose the program you just created!

PUT, GET, and APPEND are also useful for moving groups of lines in the same file. Simply put the group of lines in the save area, delete them, move to the position in the file where you want to insert them, and get them from the save area.

Add-Only Files

In addition to read-only files that can be read but not altered and read/write files that can be read and altered, there is one other type of file. An add-only file can be read but not altered except to add lines of text from the save area to the bottom of the file with the GET command. This type of file is useful for communicating with other users including the instructor. You can read notes placed in the file and reply if you wish.

Example 6

Suppose, for example, that an instructor establishes an add-only file called QA designed to be a public question/answer forum. While in SIMPLE, you can open and read the current contents of the file as follows.

Dialogue

```
>>>
.open qa
YOU MAY NOW READ AND ADD TO QA ( CONTAINS 17 LINES )
$TOP OF QA
>>>
.type 99
$TOP OF QA
     THIS FILE IS AN OPEN FORUM FOR QUESTIONS AND ANSWERS.
ASK ANY QUESTION YOU WISH.  YOU MAY ALSO ANSWER QUESTIONS
ASKED BY OTHERS.

                    DR. I.N. STRUCTOR

**************** FRED JONES 12:15 PM AUG 6, 1978 ****************

     I HAVE A QUESTION ABOUT THE PRINT STATEMENT:

          DO YOU HAVE TO PUT THE COMMA IN AFTER 'PRINT' OR CAN
          IT BE LEFT OUT?

**************** TERRY ABLE 12:23 PM AUG 6, 1978 ****************

     THE STATEMENT 'N=N+1' DOESN'T MAKE SENSE.  HOW CAN N BE
EQUAL TO ONE MORE THAN N?
$END OF QA
```

Now suppose you wish to answer Terry's question. As shown below, you need to create a file (or use an existing one), write the answer, put the answer in the save area, open QA, and get the answer out of the save area, adding it at the end of QA.

Dialogue	Explanation

```
>>>
.open note                                          Create a new file to write and, if
YOU MAY NOW EDIT NOTE ( CONTAINS 0 LINES )          necessary, edit your message.
$TOP OF NOTE
>>>
.insert
START TYPING
.terry: in fortran the equal sign does not mean equality; it
.       means 'is replaced by'.  'n = n+1' means replace the
.       current value of n by the sum of n and 1.
.
>>>
.put 99                                             If you are satisfied that your
3 LINES SAVED                                       message is in the form you want it,
>>>                                                 put it in the save area.  You will
.open qa                                            not be able to alter it after you
YOU MAY NOW READ AND ADD TO QA ( CONTAINS 17 LINES )  have transferred it to the
$TOP OF QA                                          Add-only file!
>>>
.forward 99
$END OF QA
>>>
.get
3 LINES TRANSFER INTO QA
>>>                                                 Check if your message has been added
.back 99                                            to the Question/Answer file.
$TOP OF QA
>>>
.type 99
$TOP OF QA
     THIS FILE IS AN OPEN FORUM FOR QUESTIONS AND ANSWERS.
ASK ANY QUESTION YOU WISH.  YOU MAY ALSO ANSWER QUESTIONS
ASKED BY OTHERS.

                    DR. I. N. STRUCTOR

*************** FRED JONES 12:15 PM AUG 6, 1978 ***************

     I HAVE A QUESTION ABOUT THE PRINT STATEMENT:

          DO YOU HAVE TO PUT THE COMMA IN AFTER 'PRINT' OR CAN
          IT BE LEFT OUT?

*************** TERRY ABLE 12:23 PM AUG 6, 1978 ***************

     THE STATEMENT 'N=N+1' DOESN'T MAKE SENSE.  HOW CAN N BE
EQUAL TO ONE MORE THAN N?

*************** YOUR NAME 12:35 PM AUG 6, 1978 ***************

TERRY: IN FORTRAN THE EQUAL SIGN DOES NOT MEAN EQUALITY; IT
       MEANS 'IS REPLACED BY'.  'N = N+1' MEANS REPLACE THE
       CURRENT VALUE OF N BY THE SUM OF N AND 1.
$END OF QA
```

16

Note that a banner with the name of the person who wrote the note and the time the note was placed in the file is automatically provided. From time to time, the instructor may purge the file of outdated messages.

Summary of Commands

INSERT	Insert text beginning after the current line. Insertion terminates when an empty line is entered.
DELETE n	Delete n consecutive lines beginning with the current line.
FORWARD n	Move forward n lines from the current line.
BACK n	Move back n lines from the current line.
TYPE n	Type n consecutive lines beginning with the current line.
QUIT	Quit.
HELP x	Give an explanation of command x. If x is omitted, list the currently available commands.
RUN	Run the program. The current file must contain FORTRAN code and any needed input data. The data must be placed at the end of the program and separated from the program by $DATA.
NAMES	List the names of all files available.
OPEN x	Open file x. Create a new file named x if the file x doesn't already exist.
PUT n	Clear the save area and put n consecutive lines beginning with the current line into the save area.
GET	Get the text in the save area and insert it after the current line.
APPEND n	Append n consecutive lines beginning with the current line to the save area.
LOCATE /x/	Locate the line that contains the sequence of characters x enclosed in slashes. The search begins with the line following the current line and proceeds only to the end of the file. If /x/ is omitted, locate the line that contains the sequence of characters specified most recently in a LOCATE command.
CHANGE /x/y/ n	Change occurrence of the sequence of characters x to the sequence of characters y in the next n lines beginning with the current line.

APPENDIX I - Follow-up Courses

If you enjoyed learning FORTRAN, you may want to learn another computer language or two, or learn more about computer systems in general. The language most commonly used in data processing tasks like payroll, inventory, accounting, and other transaction-oriented applications is COBOL (Common Business Oriented Language). You may learn COBOL by taking CS 238, COBOL Programming for Business (2 cr.) or, if you want a really fast-paced course, CS 252A,COBOL Programming (1 cr.). Several other one-credit programming courses are also offered for students who have already had an introductory course (most students, however, consider these quite time consuming for a one-credit course!)

Once you know both COBOL and FORTRAN, you could learn more about information organization in large files in CS-MGM 412, Data Base Organization and Management. A more technical treatment of similar topics, including the analysis of algorithms for sorting, searching, and traversing decision trees, is available in CS 310, Information Structures. Neither of these courses is a prerequisite to the other, and both may be taken for credit.

If you are interested in finding out more about how a computer actually works, consider CS 230, Computer Organization. In this course you study how instructions are stored, decoded, and executed, memory organization, internal representation of numbers and letters, and the elements of operating systems and other major software components of computer systems. Computer science students take this course in parallel with CS 231 (2 cr.), Assembler Programming. Mathematical concepts required for a deeper understanding of computer systems are covered in CS 235, Discrete Structures, which has CS 230 as a prerequisite.

Feel free to come and discuss your computer interests with the faculty and students of the Computer Science Department. A course description brochure is available in the departmental office.

In the Management Department, MGM 437 and 438 will give you a chance to test your new skills.

APPENDIX II - IBM OS VM/370-158 CP-CMS

As you work in one of the terminal rooms, some of the above jargon may strike your ears. This appendix is an attempt to put you into the large picture; however, if you are not bothered about how you relate to the computer, feel free to skip it. You are not responsible in CS 237 for any of this material.

The system

The computer you use is shared simultaneously by as many as fifty people who communicate with it through terminals. The operating system of the computer allows each user to access the central processing unit and the storage devices (disks) for brief time-slices in such a manner that each user may have the impression that he alone uses the computer. This is called time-sharing.

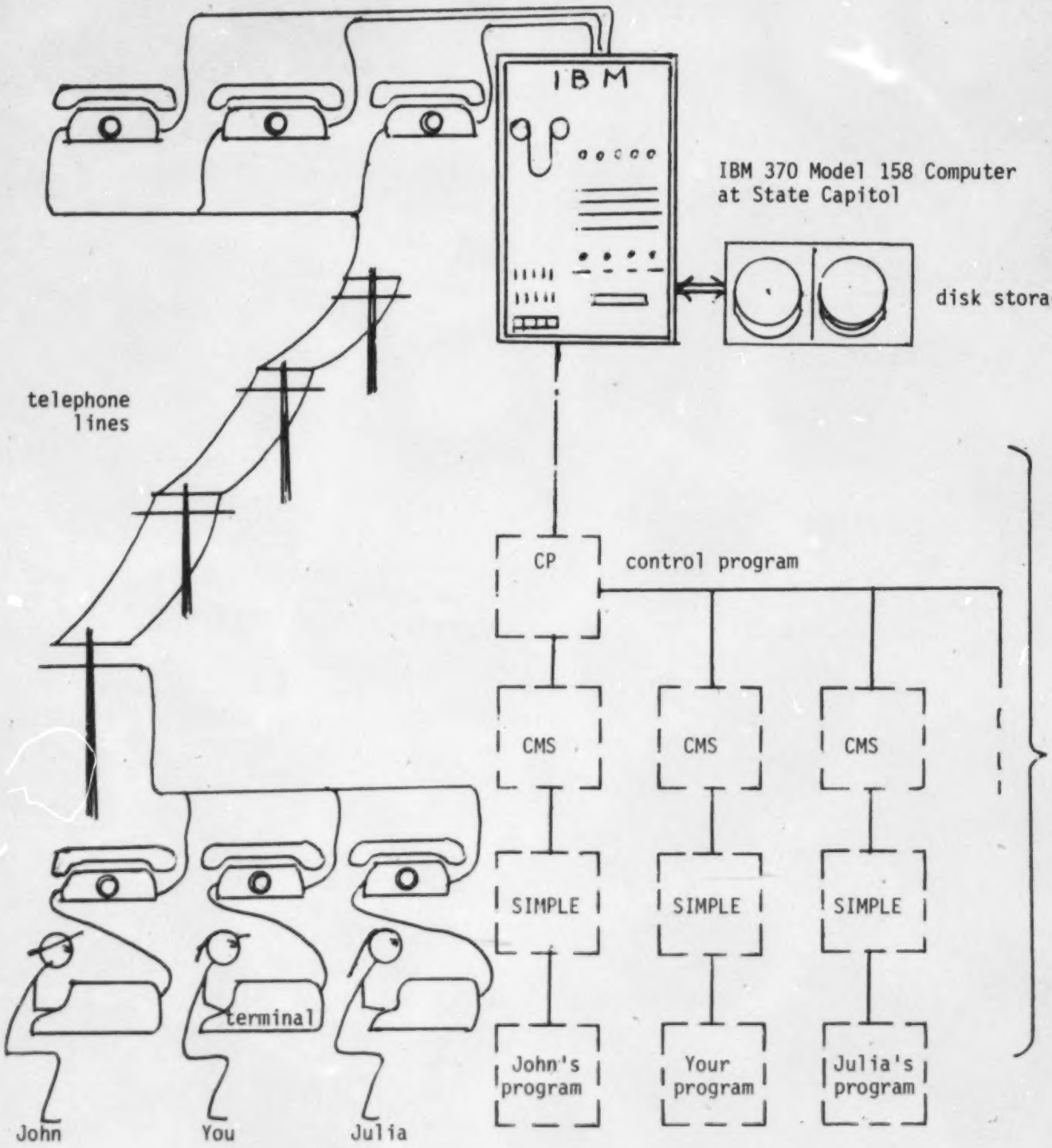
The following are a few additional details on how the operating system, which is a large and complicated program written by systems programmers at IBM, distributes computing power to each user. Do not worry if you do not fully understand exactly how this is accomplished; very few people do!

In order to let each user believe that he is in control of the whole computer, a segment of the operating system called CP (Control Program), which you reach when you dial the machine, sets up a simulation program, called a virtual machine, for you. For this reason, the entire operating system (OS) is called IBM (the manufacturer) VM (virtual machine) 370 (the type of computer) 158 (the model number): IBM VM/370 - 158.

The instructions that the computer actually executes are very elementary; add 1+1, move a number from one location in storage to another, test if a number is equal to 0. These elementary operations can, however, be combined into programs that allow users to do complicated things, like edit (change) or compile a FORTRAN program. It is also possible to query the computer to see how many people are using it or how much storage is still available. These programs are collectively known as CMS (Conversational Monitoring System). CMS contains hundreds of thousands of lines of program code. After your virtual machine is set up by CP, CP automatically provides you with a copy of the CMS programs.

CMS is very powerful, but it is still a complicated system to use for novices. The SIMPLE system for program editing and for running programs was therefore developed by the Computer Science Department to make it easier for you to take your first steps in the world of programming. After log-on, CMS puts you in the SIMPLE programming environment, and you should never have to worry about VM/370, CP, and CMS! After you have completed this course, however, you will find that it is not very difficult to make the transition to CMS. CMS will allow you to do much more than SIMPLE. In the meanwhile, perhaps the following diagram will help you understand your actual relation to the computer.

1-9575



Your Computer

The dotted boxes represent software (programs) which are usually kept on the disk storage of the computer and are transferred to the central processing unit for execution. Each program stays in the main memory of the computer only for a fraction of a second before it is moved out to make room for another program.

Computer Science 237
Week #1

Introduction to computers, SIMPLE, and the Fortran programming language

Objectives

At the end of this week you should be able to

a. Use SIMPLE to edit and run a Fortran program
b. Properly space and punctuate Fortran statements
c. Understand the rudiments of the READ, assignment, GO TO, IF, and PRINT statements

Reading Assignment

1. Read the Course Guide.

2. Read 1.1 through 1.7 in the text.
   The text introduces Fortran as if you were going to use punch cards to submit programs and data to the computer. Although you will not use cards in CS 237, you should read this material carefully. Whenever the text refers to a card, you may think of a line of text to be typed. All of the rules that tell how to punch information on a card apply directly to typing the same information on a line. See the Course Guide for further information on how to put together a Fortran package as discussed in section 1.3 of the text.
3. Read 1.8 in the text.
   You will not be tested on this material, but you should read it to at least have some notion of what is happening to your program when you submit it for execution.

Exercises

1. Read and understand problems 1.1 through 1.3 in the text.
2. Work problems 1.4 and 1.7 and check your answers with those provided in the text.
3. Write down the answers to the following questions and be prepared to discuss them as part of your first oral quiz.

   a. What is the purpose of the PRINT statement?

   b. Determine the output of the following program:

```
      M = 15
   10 IF( M .LT. 3 ) GO TO 20
      PRINT, M
      M = M-3
      GO TO 10
   20 PRINT, M
      STOP
      END
```

c. The placement of some parts of the statements in the following Fortran program is improper. Mark the statements that are wrong and state why.

```
columns: 1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

C  THIS PROGRAM TESTS YOU KNOWLEDGE OF PROPER POSITIONING OF FORTRAN STATEMENTS
      LAST = 7
     KOUNT = 1
    10 KOUNT = KOUNT + INCR
       INCR = INCR
     +      - 1
IF( INCR .LT. 0)                  GO TO !0
      PRINT,    KO     UN    T,         IN     CR
                                                                        STOP
                    E             N               D
```

Programming Assignment

Reread the first 12 pages in your Course Guide on SIMPLE.

Log-on to the computer. You will find yourself in a file named PROGRAM. It contains a working program very much like Example 1.1(c) on page 8 of your text. On the next few pages is a sample of a terminal session that manipulates the file in several ways using the SIMPLE commands RUN, FORWARD, BACK, INSERT, DELETE, LOCATE, and CHANGE. Try to follow it as closely as possible on your own terminal.

Make sure that when the program is RUN in the sample terminal session, you RUN your copy successfully too. If your program has errors, correct them and RUN the program again. If you get stuck, get help.

To be completed on time, your final program must be run by your cutoff time in the period January 22-24, as described in the Course Guide. If you do all this, you may earn up to 8 points in your first quiz. Following the sample terminal session is an extra credit assignment, PART B. If you do it also, you may earn an additional 2 points for the week.

```
Dialogue                                Comment

VM/370 ONLINE                           The computer types this when it connects.
!                                       You then press RETURN (CR).

.logon un10420                          And type your account number.
ENTER PASSWORD:
.■■■■■■■■                               Your password goes here.
LOGON AT 03:14:38 CST MONDAY 01/08/79
CMS 5.04C
.                                       Press CR again.
A (191) R/O
PLEASE TYPE IN YOUR SIMPLE NUMBER.
■■■■■■                                  Last 4 digits of your SS #.

PLEASE WAIT, SIMPLE IS BEING READIED FOR YOU.

ENTER YOUR FIRST NAME                   Only asked the first time you logon.
.willie
ENTER YOUR LAST NAME
.wonka

YOU MAY NOW EDIT PROGRAM (CONTAINS 9 LINES)
$TOP OF PROGRAM                         This is the beginning of PROGRAM.
>>>.type 99                             Look at contents first.
$TOP OF PROGRAM
C
C  CALCULATE THE AREA OF A TRIANGLE GIVEN ITS BASE, B, AND HEIGHT, H.
C
      B = 10.0
      H =  8.0
      AREA = (B*H)/2.0
      PRINT, B, H, AREA
      STOP
      END
$END OF PROGRAM
>>>.run                                 Execute program in the file.
PROGRAM HAS BEEN
SUBMITTED FOR EXECUTION - STAND BY FOR RESULTS.

         $JOB
         C
         C  CALCULATE THE AREA OF A TRIANGLE GIVEN ITS BASE, B, AND HEIGHT, H.
         C
     1         B = 10.0
     2         H =  8.0
     3         AREA = (B*H)/2.0
     4         PRINT, B, H, AREA
     5         STOP
     6         END

        $ENTRY
  0.1000000E 02    0.8000000E 01    0.4000000E 02  This is the output.
                                    (Real numbers look funny, what are they?)
CORE USAGE       OBJECT CODE=     288 BYTES,ARRAY AREA=       0 BYTES,TOTAL AI

DIAGNOSTICS        NUMBER OF ERRORS=       0, NUMBER OF WARNINGS=       0, NU

COMPILE TIME=     0.03 SEC,EXECUTION TIME=     0.00 SEC,  WATFIV - JUL 1973 V

         $STOP
```

```
YOU MAY NOW EDIT PROGRAM (CONTAINS 9 LINES)
$TOP OF PROGRAM                 File ready for manipulation.
>>>.forward 4                   Go past $TOP and first 3 lines.
      B = 10.0                  We are now located on this line.
>>>.delete 2                    Get rid of it and one following it.
TO DELETE 2 LINES TYPE Y, OTHERWISE TYPE N    Just checking!
.y                              It is easy to destroy lines
2 LINES DELETED                  you really want to keep.
C                               Line just above 'B = 10.0'.
>>>.insert                      We want to put in something else.
START TYPING
.      read, b, h               Let's READ our data.
.                               An empty line stops insertion.
>>>.forward 99                  Go to end of file.
$END OF PROGRAM
>>>.insert                      Put in data to be read.
START TYPING
.$data                          $DATA signals its beginning.
. 10.0  8.0
.                               Remember to stop inserting.
>>>.back 99
$TOP OF PROGRAM
>>>.type 99                     Look at what we've done.
$TOP OF PROGRAM
C
C  CALCULATE THE AREA OF A TRIANGLE GIVEN ITS BASE, B, AND HEIGHT, H.
C
      READ, B, H
      AREA = (B*H)/2.0
      PRINT, B, H, AREA
      STOP
      END
$DATA
 10.0  8.0
$END OF PROGRAM
>>>.run                         O.K.?  Then execute it.
PROGRAM HAS BEEN
SUBMITTED FOR EXECUTION - STAND BY FOR RESULTS.

         $JOB
         C
         C  CALCULATE THE AREA OF A TRIANGLE GIVEN ITS BASE, B, AND HEIGHT, H.
         C
     1         READ, B, H
     2         AREA = (B*H)/2.0
     3         PRINT, B, H, AREA
     4         STOP
     5         END

         $ENTRY
   0.1000000E 02    0.8000000E 01    0.4000000E 02
                                Different method, same result.
CORE USAGE        OBJECT CODE=     296 BYTES,ARRAY AREA=       0 BYTES,TOTAL A

DIAGNOSTICS         NUMBER OF ERRORS=       0, NUMBER OF WARNINGS=       0, NU

COMPILE TIME=     0.02 SEC,EXECUTION TIME=     0.00 SEC,  WATFIV - JUL 1973 V

         $STOP
```

```
YOU MAY NOW EDIT PROGRAM (CONTAINS 10 LINES)
$TOP OF PROGRAM
>>>.forward 4                    Let's use more descriptive
      READ, B, H                 variable names.
>>>.change
CHANGE WHAT?                     'B' should be 'BASE'.
.b
TO WHAT?
.base                            Line as changed:
      READ, BASE, H
>>>.change /h/height/            'H' should be 'HEIGHT'.
      READ, BASE, HEIGHT         Now line looks like this:
>>>.forward
      AREA = (B*H)/2.0           There are still 2 lines to change.
>>>.change /b/base/2             We'll change 2 lines at once.
      AREA = (BASE*H)/2.0        See?
      PRINT, BASE, H, AREA
>>>.change /h/height/2           Want to see that again?
      AREA = (BASE*HEIGHT)/2.0
      PRINT, BASE, HEIGHT, AREA
>>>.back 99
$TOP OF PROGRAM
>>>.type 99                      Let's see what we've done.
$TOP OF PROGRAM
C
C  CALCULATE THE AREA OF A TRIANGLE GIVEN ITS BASE, B, AND HEIGHT, H.
C                                Oops, comment doesn't fit any more.
      READ, BASE, HEIGHT
      AREA = (BASE*HEIGHT)/2.0
      PRINT, BASE, HEIGHT, AREA
      STOP
      END
$DATA
 10.0  8.0
$END OF PROGRAM
>>>.forward 2                    Find that line.
C  CALCULATE THE AREA OF A TRIANGLE GIVEN ITS BASE, B, AND HEIGHT, H.
>>>.change
CHANGE WHAT?
., b,                            And make ', B,'
TO WHAT?
.                                into nothing!
C  CALCULATE THE AREA OF A TRIANGLE GIVEN ITS BASE AND HEIGHT, H.
>>>.change /, h//                This is another way to do that.
C  CALCULATE THE AREA OF A TRIANGLE GIVEN ITS BASE AND HEIGHT.
>>>.run                          Shouldn't need to be rechecked
PROGRAM HAS BEEN                    before executing it.
SUBMITTED FOR EXECUTION - STAND BY FOR RESULTS.

        $JOB
        C
        C  CALCULATE THE AREA OF A TRIANGLE GIVEN ITS BASE AND HEIGHT.
        C
     1        READ, BASE, HEIGHT      Clearer, isn't it?
     2        AREA = (BASE*HEIGHT)/2.0
     3        PRINT, BASE, HEIGHT, AREA
     4        STOP
     5        END

        $ENTRY
  0.1000000E 02    0.8000000E 01    0.4000000E 02

CORE USAGE       OBJECT CODE=     296 BYTES,ARRAY AREA=       0 BYTES,TOTAL

DIAGNOSTICS        NUMBER OF ERRORS=  0, NUMBER OF WARNINGS=       0,
```

```
YOU MAY NOW EDIT PROGRAM (CONTAINS 10 LINES)
$TOP OF PROGRAM                    Now let's calculate several areas
>>>.forward 3                         in our program.
C
>>>.insert
START TYPING
.       kount = 0                  Need to count how many we do.
.                                  Remember to get out of INSERT.
>>>.locate /print/                 Find the line that says 'PRINT'.
      PRINT, BASE, HEIGHT, AREA
>>>.insert
START TYPING
.       kount = kount + 1          Count the calculation, and go back
.       if( kount .lt. 4 ) go to 10                    three times.
.                                  Remember the CR.
>>>.back 2                         Go back to beginning of calculation.
      READ, BASE, HEIGHT
>>>.change /   read/10 read/       Put in a statement number.
   10 READ, BASE, HEIGHT
>>>.locate /10.0/                  Go down into the data.
 10.0  8.0                            (this is faster than counting lines)
>>>.insert
START TYPING                      .Put in three more bases and heights.
.  8.0 10.0
. 38.0  2.0
.126.0 95.0
.
>>>.back 99                        Now let's see what we've done.
$TOP OF PROGRAM
>>>.type 99
$TOP OF PROGRAM
C
C  CALCULATE THE AREA OF A TRIANGLE GIVEN ITS BASE AND HEIGHT.
C
      KOUNT = 0
   10 READ, BASE, HEIGHT
      AREA = (BASE*HEIGHT)/2.0
      PRINT, BASE, HEIGHT, AREA
      KOUNT = KOUNT + 1
      IF( KOUNT .LT. 4 ) GO TO 10
      STOP
      END
$DATA
 10.0  8.0
  8.0 10.0
 38.0  2.0
126.0 95.0
$END OF PROGRAM
```

```
>>>.run                          O.K.?  Run it.
PROGRAM HAS BEEN
SUBMITTED FOR EXECUTION - STAND BY FOR RESULTS.

        $JOB
        C
        C  CALCULATE THE AREA OF A TRIANGLE GIVEN ITS BASE AND HEIGHT.
        C
     1        KOUNT = 0
     2     10 READ, BASE, HEIGHT
     3        AREA = (BASE*HEIGHT)/2.0
     4        PRINT, BASE, HEIGHT, AREA
     5        KOUNT = KOUNT + 1
     6        IF( KOUNT .LT. 4 ) GO TO 10
     7        STOP
     8        END

        $ENTRY
   0.1000000E 02     0.8000000E 01     0.4000000E 02
   0.8000000E 01     0.1000000E 02     0.4000000E 02
   0.3800000E 02     0.2000000E 01     0.3800000E 02
   0.1260000E 03     0.9500000E 02     0.5985000E 04
                              Voila!
CORE USAGE        OBJECT CODE=      384 BYTES,ARRAY AREA=       0 BYTES,TOTAL AREA

DIAGNOSTICS         NUMBER OF ERRORS=         0, NUMBER OF WARNINGS=       0, NUMBEI

COMPILE TIME=      0.04 SEC,EXECUTION TIME=      0.02 SEC, WATFIV - JUL 1973 V1L4
                              Fast computer - -  - ↑
        $STOP

YOU MAY NOW EDIT PROGRAM (CONTAINS 16 LINES)
$TOP OF PROGRAM
>>>.quit                              Remember to quit while you're ahead!
TERMINAL SESSION LASTED 13 MINUTES AND COST $.61
YOU HAVE $94.81 LEFT IN YOUR ACCOUNT
CONNECT= 00:12:51 VIRTCPU= 000:02.91 TOTCPU= 000:07.85
LOGOFF AT 01:17:35 CST MONDAY 01/08/79
                              Wait for this stuff to print out!
VM/370 ONLINE
```

PART B:

If done correctly, this assignment is worth up to 2 additional points in your first oral quiz.

Make the following changes to the final program from the sample terminal session:

a. In addition to calculating the area of each triangle, calculate the area of a rectangle with the same base and height as the triangle, and print that number on a line separate from the other three numbers you printed. You will need another assignment statement and a PRINT statement to do this.

b. Instead of having your program calculate 4 sets of data, make it do these calculations for 5 sets of data. The added set will be:
   BASE:425.0, and HEIGHT:377.0.
   This will force you to make changes to both your data and your IF statement.

WEEK #2

FORTRAN Arithmetic Expressions, Assignment Statements, and Computer Arithmetic

Objectives

At the end of this week you should be able to

a. Recognize FORTRAN constants and variable names and classify them as being either of INTEGER or REAL type

b. Evaluate FORTRAN arithmetic expressions

c. Write algebraic expessions in FORTRAN

d. Explain how the assignment statement works

Reading Assignment

1. Read 2.1 through 2.9 in the text.

2. Read 2.10 in the text.
   You will not be tested on this material, but you should read it to at least have some notion of the major pitfalls of computer arithmetic.

3. Reread the course manual. You will be quizzed about using SIMPLE.

Exercises

1. Read and understand problems 2.2, 2.4, 2.8, 2.12, 2.13 and 2.14 in the text.

2. Work problems 2.24, 2.27, 2.32 and 2.34 and check your answers with those provided in the text.

3. Write down the answers to the following questions and be prepared to discuss them as part of your weekly oral quiz.

   a. Below is a list of proposed INTEGER variable names. Mark those that are improper FORTRAN variable names and explain why. For those variable names that would normally be REAL by the first letter rule, give a type statement that would allow you to use them as INTEGER variables in a program.

      COUNT
      AVERAGE
      NR_RCVD
      9TY
      NR RCVD
      N

b. What is the value of X, Y, Z, M and N after the following FORTRAN program is executed?

```
X= 3.0
Y= 2.0
M= 4
N= 5
Z= X+Y
Y= M*N + 1
M= M/N -3
N= Z/X + 0.75 * N
```

X= Y= Z= M= N=

c. Write a FORTRAN expression to compute the monthly house payment M given the principal amount of the mortgage P, the monthly interest rate I, and the number of monthly payments N.

These quantities are related by the formula

$$M = \frac{PI(1+I)^N}{(1+I)^N - 1}$$

Programming Assignment

You have a choice of any of four problems:

1. OPTION I, PART A
2. OPTION I, PART B
3. OPTION II, PART A
4. OPTION II, PART B

For both options, part B is more difficult and is worth two additional points. Your program must be run by the time assigned.

OPTION I :

You may choose either Part A or Part B in this option. Part A is worth 8 points, Part B is worth 10 points.

>>>>>>PART A :

CONVERSION FROM CENTIGRADE (CELSIUS) TO FAHRENHEIT

You must do both section 1 and section 2 of this assignment (first do section 1).

SECTION 1 :
Fahrenheit and centigrade are the names of two different scales for measuring temperature. If we let Fahrenheit degrees be FAHR and centigrade degrees be CENT, then the relation between these scales is defined by the following formula :

$$FAHR = \frac{9 * CENT}{5} + 32$$

Your task is to write a program that uses the above formula to convert 20 degrees centigrade to its equivalent in degrees Fahrenheit.
You should use real arithmetic in your program; see sections 2.2 and 2.3 of your text to review real arithmetic.

OUTPUT :
You should print one line with degrees centigrade and its equivalent in Fahrenheit respectively.
Output will automatically be printed in E format; read section 2.2 page 20 of your text for exponential form.

If CENT=20 then the output should be :

0.2000000E 02 0.6800000E 02

HINTS :
Use the above formula and write it as a FORTRAN expression(to see how it is done, look at example 2.8 on page 25 of your text). After computing Fahreheit degrees you need to print both of them (centigrade and Fahreheit), so you should use a print statement like the one in example 1.1 on page 8 of your text. If you run the program with a mistake in it the computer will give you one or more error messages. Correct the error(s), and run your program again.

SECTION 2 :
Modify the program you have written for section 1 to be able to convert degrees centigrade from 0 through 100 (incrementing by 5) to degrees Fahrenheit.
Your output should look like this

```
0.0000000E 00    0.3200000E 02
0.5000000E 01    0.4100000E 02
      .                .
      .                .
0.1000000E 03    0.2120000E 03
```

HINTS

If we have a variable A and we want to print it 5 times, instead of writing the print statement 5 times, we can use the IF and GO TO statements as shown below :

```
      I=0
10    PRINT,A
      I=I+1
      IF(I.LT.5) GO TO 10
      STOP
      END
```

>>>>>>PART B :

CALCULATION OF COMPOUND INTEREST

You must do both section 1 and section 2 of this assignment (first do section 1).

SECTION 1 :

If you invest $1000 at a 10 percent interest rate, after one year you will have $1100 (1000+10 percent of 1000). You can repeat this computation year after year as follows:

```
1 year     1000+100=1100
2 years    1100+110=1210
3 years    1210+121=1331
   .           .
   .           .
   .           .
```

If you want to know how much an amount A at interest rate I will be worth after Y years, the following formula allows you to calculate the total T :

$$T=A(1+I/100)^{Y}$$

You are to write a program to compute this total amount T. Your program should assign the value 30000.00 to the variable A, the value 10 to the variable I and 10 to the variable Y.

Amount A is expressed in dollars and cents, interest I is a real number and number of years Y is an integer.
To review the concept of real and integer numbers read section 2.2 of your text.

OUTPUT :

Your program should generate one output line with amount, interest rate, number of years and total amount respectively.
Thus for above values the output is :

```
0.3000000E 05        0.1000000E 02        10        0.7781227E 05
```

Output will automatically be printed in E format. Read section 2.2 on page 20 of your text for exponential form.

HINTS :

First you should use assignment statements to assign values to A, I and Y. Then compute the total using the above formula written as a FORTRAN expression. (Study example 2.8 on page 25 of the text) Use a PRINT statement like the one in example 1.1 on page 8 of the text to print the results. If you run the program with a mistake in it, the computer will type an error message. Correct the error and run your program again.

SECTION 2 :

Modify and run the program you have written for SECTION 1, to be able to compute and print the total for years in the range of 0 to 30 in steps of 5 years.

HINT :

See hint for part A of this Option.

OPTION II:

You may choose either Part A or Part B as your assignment. Part A is worth 8 points, Part B is worth 10 points.

PART A:

CONVERSION OF SQUARE YARDS TO ACRES

You must do both section 1 and section 2 of this assignment. Do section 1 first.

Section 1

An acre contains 4,840 square yards. If we use the variable name YARD for square yards and ACRE for acreage, then we can express the relation between them as

$$ACRE = \frac{YARD}{4850}$$

Write a program that will convert 10,000 square yards to the equivalent measure in acres. You are going to modify this program in Section 2 to print the acreage for several values of YARD so in your program you should assign YARD the value 10,000. You should use REAL variables and REAL constants in you program. Review sections 2.2 and 2.3 in your text. You will also need to convert the formula given above into a FORTRAN assignment statement. Don't forget to include the PRINT statement. Your program will be similar to the one on page 6 of your text.

If your program contains one or more errors the computer will print error messages to help you find the errors. Correct your program and run it again. If your program is correct the output the computer prints will look like this

```
 0.1000000E 05    0.2061855E 01
```

Section 2

Change the program you wrote for Section 1 so it will convert square yards in the range of 10,000 to 100,000 in steps of 10,000 to their equivalent in acres. Your output should look like this

```
 0.1000000E 05     0.2061855E 01
 0.2000000E 05     0.4123711E 01
      .                 .
      .                 .
 0.1000000E 06     0.2061855E 02
```

HINT:

Study examples on pages 9 and 15 of your text. You will need a statement such as

IF(ACRE .LT. 100001.0) GO TO ___

When the computer comes to this statement in your program it will go to the statement whose label you have put in the blanks. Review section 1.6 on page 8 of your text if you do not understand this.

PART B

The manager of JAKES SHOE STORE wants to raise all the prices in the store by multiplying by 1.15. ('Crazy' Jake plans to slash all prices in the greatest sale ever next month.) All the prices in the store end in the magic number .49. Write a program that will print a list of the old prices between $10.49 and $25.49 and beside each print the corresponding new price. Note that the new price will not necessarily end in .49. In your program let all prices be in cents and use integer arithmetic. You may round down (that is truncate) any fractions of cents. Your output should also be in cents. To review integer arithmetic and how to use it to truncate fractions see page 22 in your text. In our case we need to first multiply by 115 and then divide by 100 (notice no decimals here). We might use a FORTRAN statement like

```
NEWPR = OLDPR * 115/100
```

The output the computer prints should look like this

```
1049        1206
1149        1321
  .           .
  .           .
2549        2931
```

If you make one or more errors in your program the computer will print error messages. Correct you errors and run your program again.

EOF:

WEEK #3 (September 14 - September 20)

Input/Output

Objectives

At the end of this week you should be able to

a. Understand the difference between formatted and unformatted input/output

b. Explain the operation of the format codes I, F, and X for both input and output

c. Determine the carriage control specification of a FORMAT statement

d. Use literals to appropriately label computer output

Reading Assignment

1. Read 3.1 in the text.

2. Read 3.2 in the text.

   On our system, both commas and blanks may be used to separate numbers in input data for unformatted READ statements.

3. Read 3.3 in the text.

   On our system, 5 is the input unit number and 6 is the output unit number.

4. Read 3.4 in the text skipping the section on E-fields.

5. Read 3.5 in the text.

   On our system, the carriage control character must be present, but is otherwise ignored. The system always advances one line even if the carriage control specifies something else. Hence, it is best if you use 1X for carriage control in all FORMAT statements. For the quiz, however, you must understand how the carriage control character works for line printers.

6. Read 3.6 in the text skipping the section on E-fields.

7. Read 3.7 in the text.

Exercises

1. Read and understand problems 3.2, 3.4, 3.6a, 3.6b, 3.7, and 3.14 in the text.

2. Work problems 3.22, 3.26a, 3.27, and 3.32f and check your answers with those provided in the text.

3. Write down the answers to the followins questions and be prepared to discuss them as part of your weekly oral quiz.

   a. Given the input data as shown:

```
                                  1         2
      card columns:  12345678901234567890

                     1234   94.27  219
```

   What is the output when the followins FORTRAN program sesment is executed?

```
      READ(5,10) I,Y,Z
   10 FORMAT(2X,I3,F7.2,F5.1)
      WRITE(6,20) Y,Z,I
   20 FORMAT(1X,'THE OUTPUT IS:',F8.1,F6.1,3X,I3)
```

   b. For each of the followins FORMAT statements, determine the carriage control character and explain what action is specified.

```
      FORMAT('1','PAGE ',I3)
      FORMAT('0','X= ',F10.7)
```

Programmins Assignment

Choose ans one of four problems:

1. Option I, Part A
2. Option I, Part B
3. Option II, Part A
4. Option II, Part B

For both options, part B is more difficult and is worth two additional points. To be completed on time, your program must be run by the time assisned.

OPTION I

You may choose either A or B below as your assignment. A is worth 8 points and B is worth 10 points.

>>>>>>PART A :

CONVERSION FROM CENTIGRADE (CELSIUS) TO FAHRENHEIT

Modify the program you wrote last week (OPTION I, PART A-2), but this time your output should be appropriately labeled and the values printed in F format. Use a FORMAT statement for output.
Read sections 3.6 page 52 and 3.7 page 54 (literal field).

Now your printout should look like this:

CONVERSION TABLE FROM CENTIGRADE TO FAHRENHEIT

CENTIGRADE	FAHRENHEIT
0.0	32.0
5.0	41.0

etc.

>>>>>>PART B :

CALCULATION OF COMPOUND INTEREST

Do the program you did last week (OPTION I, PART B-2), but this time the output must be neatly labelled and the values printed using appropriate F and I FORMAT codes. Read sections 3.6 on page 52 and 3.7 on page 54.
Your output should look something like this.

CALCULATION OF COMPOUND INTEREST

AMOUNT	INTEREST RATE	YEAR	TOTAL
$30000.00	10.0%	10	$77812.27

etc.

OPTION II

YOU MAY CHOOSE A OR B BELOW AS YOUR ASSIGNMENT.
A IS WORTH 8 POINTS AND B IS WORTH 10 POINTS.

PART A:
Change the program you wrote for last week (OPTION II, PART A-2). This time your output must be appropriately labelled and the values printed in F format. Use FORMAT statements for your output. Read sections 3.6 and 3.7 pages 52 and 54 in your text.
Your output should look like this

CONVERSION OF SQUARE YARDS TO ACRES

SQUARE YARDS	ACRES
10000.	2.06

PART B:

WINNIE DOWN IN THE KEY PUNCH DEPARTMENT PLAYS GIN RUMMY WITH SOME OF HER FRIENDS EVERY MORNING FOR COFFEE BREAK. (NOTICE THAT ALL THE CARDS REFERRED TO FROM HERE ON ARE PUNCHED CARDS NOT PLAYING CARDS THOUGH.) SHE WOULD LIKE YOU TO WRITE A PROGRAM TO BE RUN EACH DAY THAT WILL PRINT OUT THE NET WINNINGS OF EACH OF THE FOUR PLAYERS. SHE DOESN'T MIND PUNCHING CARDS, OF COURSE, AND WOULD PUNCH TWO CARDS EACH DAY, ONE WITH THE NET WINNINGS COMPUTED THE PREVIOUS DAY AND A SECOND WITH THE CURRENT DAYS WINNINGS. (IT ISN,T THAT WINNIE CAN'T ADD, IT'S THAT SHE TENDS TO WIN AND LIKES TO SEE THE RECORD IN PRINT.)

SO WRITE A PROGRAM USING FORMATTED INPUT/OUTPUT (YOU EXPECTED THAT) WHICH WILL READ HER TWO CARDS AND PRINT OUT THE NEW NET WINNINGS. (BY 'NET WINNINGS' WE MEAN A RUNNING TOTAL OF THE AMOUNT WON OR LOST. THUS, IF THE NET WINNINGS OR RUNNING TOTAL WAS -20.00 YESTERDAY AND 4.00 WAS WON TODAY, THE NEW NET WINNINGS WOULD BE -16.00.) SPECIFICALLY, YOUR PROGRAM SHOULD:

1. READ THE CARD WITH YESTERDAY'S NET WINNINGS,
2. READ THE CARD WITH TODAY'S WINNINGS,
3. FOR EACH PLAYER, ADD TODAY'S WINNINGS TO YESTERDAY'S NET WINNINGS, AND
4. PRINT THE SUMS AS TODAY'S NET WINNINGS.

NOTE THAT SOME OF THESE NUMBERS WILL BE NEGATIVE. SUPPOSE, FOR EXAMPLE, THAT YESTERDAY'S NET WINNINGS WERE:

```
WINNIE   43.56
JENNIE  -23.56
FANNIE  -18.45
ANNIE    -1.55
```

AND THAT TODAY'S WINNINGS ARE:

```
WINNIE   -5.26    (SHE LOST!!!)
JENNIE    1.00
FANNIE    3.00
ANNIE     1.26
```

THE NEW NET WINNINGS WOULD THEN BE:

```
WINNIE   38.30
JENNIE  -22.56
FANNIE  -15.45
ANNIE     -.29
```

YOU MAY TELL WINNIE WHAT FORMAT TO USE WHEN PUNCHING THE NUMBERS IN HER CARDS.

YOU SHOULD PRINT A HEADING FOR THE OUTPUT AND ALSO PRINT EACH PERSONS PERSON'S NAME ABOVE HER WINNINGS. PRINT YESTERDAYS NET WINNINGS TODAYS RESULTS AND THE CURRENT WINNINGS. YOUR OUTPUT SHOULD LOOK SOMETHING LIKE THIS:

GIN RUMMY WINNINGS

	WINNIE	JENNIE	FANNIE	ANNIE
YESTERDAYS TOTAL	43.56	-23.56	-18.45	-1.55
TODAYS RESULT	-5.26	1.00	3.00	1.26
TODAYS TOTAL	38.30	-22.56	-15.45	-.29

HAVING TROUBLE GETTING STARTED? LOOK AT THE STATEMENTS IN EXAMPLE 3.2 PAGE 46. SINCE YOU HAVE TWO CARDS TO READ, YOU MIGHT AS WELL (FOR NOW) USE TWO RLAD STATEMENTS. SO DECIDE ON NAMES FOR THE VARIABLES TO BE READ AND WRITE THE READ STATEMENTS LIKE THE ONE IN THE EXAMPLE. IF YOU USE THE SAME FORMAT ON BOTH CARDS YOU ONLY HAVE TO WRITE ONE FORMAT STATEMENT, BUT YOU'LL NEED TO USE ITS LABEL IN BOTH READ STATEMENTS. NOW YOU NEED TO PICK THE FIELD WIDTH TO GO WITH EACH NUMBER ON THE CARDS, FOR INSTANCE, 7 WIDE FOR EACH. YOU'RE READY TO WRITE THE FORMAT STATEMENT. FOLLOW THE EXAMPLE ON PAGE 46, ONLY YOU WILL NEED 4 FIELD SPECIFICATIONS LIKE F7.2 .

REMEMBER TO HAVE YOUR PROGRAM WRITE THE HEADING AND THE NAMES OF THE WINNERS. YOU SHOULD ALSO WRITE A LINE SHOWING YESTERDAYS TOTAL.

NEXT WRITE 4 STATEMENTS, EACH OF WHICH FINDS THE NEW NET WINNINGS FOR ONE OF THE 4 PLAYERS. AFTER THIS YOU NEED TO HAVE A WRITE STATEMENT AND A FORMAT STATEMENT LIKE THE ONES ON PAGE 52. WHY NOT PICK A WIDER FIELD FOR THE OUTPUT, SAY F20.2 .

DON'T FORGET TO TYPE IN YOUR TWO DATA CARDS AFTER THE END OF YOUR PROGRAM. TYPE THEM ACCORDING TO THE FORMAT YOU DECIDED ON ABOVE IN THE READ FORMAT STATEMENT.

REVIEW THE ASSIGNMENT YOU DID FOR WEEK 1. DON'T FORGET TO USE THE $DATA CARD.

Computer Science 237

WEEK #4

Flow of control

Objectives

At the end of this week you should be able to

a. Convert a flowchart to FORTRAN code

b. Write relational expressions in FORTRAN

c. Use logical IF statements to selectively execute segments of code and to control loops

d. Explain the use of header and trailer cards.

Reading Assignment

1. Read 4.1 through 4.6 in the text.

2. Read 4.7 and 4.8 (you will not be responsibe for this material).

3. Read 4.9 and 4.10 in the text.

4. Review the section of the course outline entitled: 'Working with more than 1 file'. (Page 33)

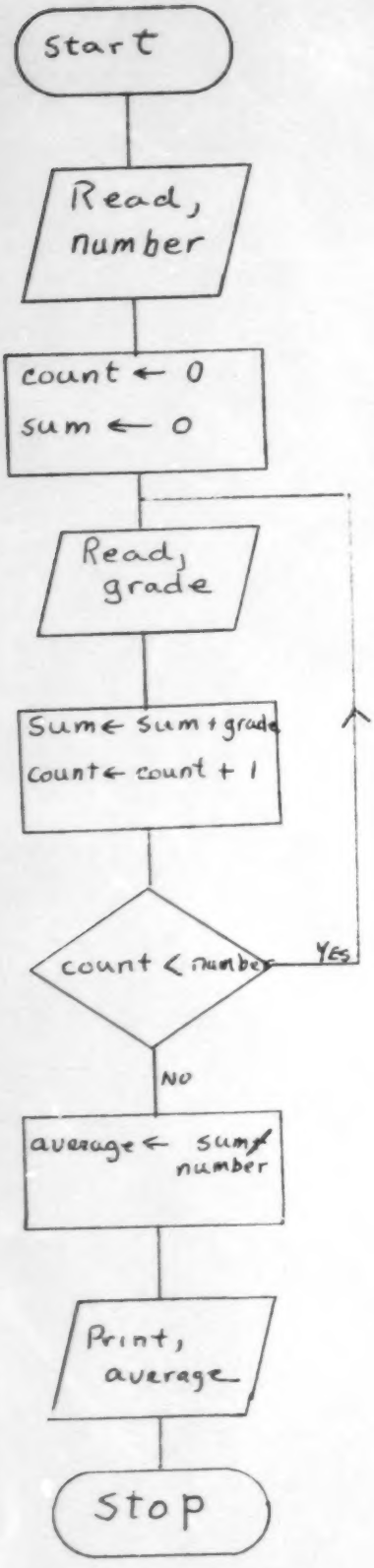
Exercises

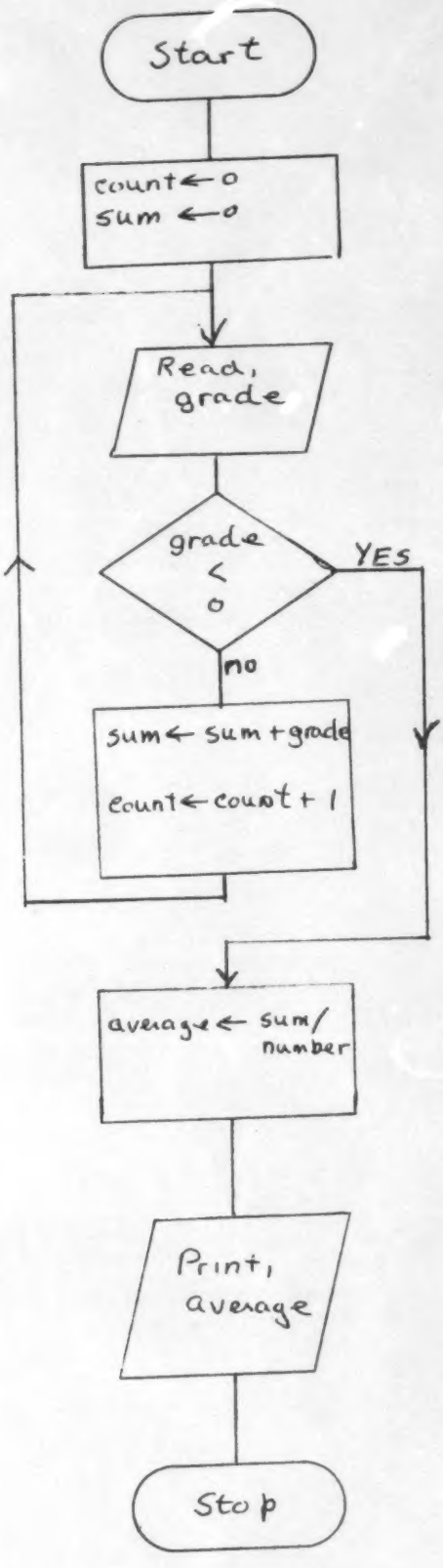
1. Read and understand problems 4.1, 4.3, 4.6a, 4.6e, 4.6f, 4.9, 4.12

2. Write down the answer to question 4.21 and check you answer with the answer in the text.

3. Write down the answers to the following questions and be prepared to discuss them at your oral quiz.

a. Write the FORTRAN code for the following flowcharts.


Start
Read, number
count ← 0
sum ← 0
Read, grade
Sume← Sum + grade
count← count + 1
count < number
YES
NO
average ← sum/ number
Print, average
Stop


Start
count← 0
sum ← 0
Read, grade
grade < 0
YES
no
sum← sum + grade
count← count + 1
average ← sum/ number
Print, average
Stop

Choose any one of four problems:

1. Option I, Part A
2. Option I, Part B
3. Option II, Part A
4. Option II, Part B

For both options, part B is more difficult and is worth two additional points. To be completed on time, your program must be run by the time assigned.

For all four problems this week you will need to use the SIMPLE commands: open, put and get.
These commands are used to add the data file provided to the end of your program. A possible sequence of commands to do this is:

```
1. open data        (replace 'data' with name of data file )
2. put 99           (put all the data lines into the savearea)
3. open prog        (replace 'prog' with name of your file)
4. forward 100      (go to end of file )
5. get              (lines saved added to your file)
```

Remember to insert the $DATA line between your program and the data.

OPTION I

You may choose either A or B below as your assignment. A is worth 8 points and B is worth 10 points.

>>>>>>PART A:

COMMISSION CALCULATION

A salesman gets a $7.50 commission on each sale. If the sale is over $150.00 he gets an additional bonus of 3 percent of the amount over $150.00. For example, if the sale is $300, he is $150 over the $150 limit, therefore he gets 3 percent of $150 or a $4.50 bonus; this makes his total commission $12. If the sale is $400, then he is $250 over the limit, so he gets an extra $7.50 or a total of $15.

You are to write a program that will read the amounts of sales and print the amount and the commission for each sale.

INPUT :
Each data card has a number in columns 1-10, showing the amount of a sale. The amount is expressed in dollars and cents, so you should use two decimal places for each number. The number of data cards is unknown but the last card in the deck has a 0 (zero) for the amount of sale to signal the end of the data cards; this is called a trailer card and is used to help the programmer terminate the execution of the program when encountered (read section 4.10 page 95 of your text).
The data for this program is in file D4IA.

SAMPLE CARD :

```
columns:        1       9
  data :            1150.00
```

OUTPUT :
You should print the amount and the corresponding commission for each sale. Your output should be appropriately labeled.
With the above sample card the output should be :

RESULTS OF COMMISSION CALCULATION FOR A SALESMAN

AMOUNT	COMMISSION
1150.00	47.50

You should use FORMAT statements for both input and output (read sections 3.4, 3.6, and 3.7 of your text)

HINTS :
You know that if the amount is less than or equal to $150 the commission is equal to $7.50 but if it is greater than $150 you should compute the commission according to the above specifications.
You can assign 7.50 to the variable you have set up for the commission and check to see whether amount is less than or equal to $150. If it is you want to go to the part of the program to print the amount and commission, but if it is greater than $150, you want to first compute the commission and then print the result. You can make this decision using the IF and GO TO statements in the following form :

```
          ................
          ................
          IF(AMOUNT .LE. 150.) GO TO 20
          .......(here you want to compute the commission)
          ................
          ................
   20     .......(here you want to print the result)
          ................
          ................
```

You can also read sections 4.2 through 4.5 for more detail.
Notice that after reading the amount your program must check for a zero value to stop the execution of the program.

>>>>>>PART B :

CALCULATING THE AMOUNT OF INCOME TAX

In a country (somewhere in the world!) the income subjected to tax is calculated by taking a person's yearly salary and subtracting 10 percent of this salary as a standard deduction then subtracting $750 for each dependent. So if a person with 6 dependents has a yearly salary of $10000, the amount subjected to income tax (AMOUNT) is :

AMOUNT=10000-(0.10*10000)-(6*750)=4500 dollars

10 percent of this amount is the amount of tax the person should pay. However if the sum of these deductions is greater than his annual salary (AMOUNT is negative) then he is considered 'low income' and does not pay any tax (and should actually receive some money!). If AMOUNT is greater than 10000 dollars then the tax is 10 percent of the first 10000 plus 15 percent of the amount over 10000.

Write a program to read the yearly salary and number of dependents of a person. Compute the amount of income tax or report that the person is a low income person.

INPUT :

Each data card has two numbers in the following format :

columns	data
1-10	annual salary (with two decimal places)
13-14	number of dependents (integer)

SAMPLE CARDS:

```
columns :    1        9
   data :        9000.50   5
   data :        3000.00   7
```

The number of cards is unknown, but there is a negative number in the salary field of the last card to signal the end of the data cards. This card is called a trailer card and is used to terminate the execution of the program when encountered. (Read section 4.10 page 95 of your text.) The data for this program is in a file called D4IB.

OUTPUT :

Your output should be appropriately labeled and you should print the salary, no. of dependents and amount of income tax for each person. In the case of low income persons instead of the amount of tax, print 'LOW INCOME'.

With the above sample cards your output should be :

CALCULATING THE AMOUNT OF INCOME TAX

SALARY	NO. OF DEPENDENTS	TAX
9000.50	5	435.04
3000.00	7	LOW INCOME

Use FORMAT statements for all input and output. See sections 3.4, 3.6 and 3.7 of your text.

HINTS :

After computing the amount subject to income tax, you need to know whether it is negative. If it is then you want to go to the segment of your program where you print salary, number of dependants and the words 'LOW INCOME'. If amount is positive then you need to find out if it is greater than 10000, if it is not you compute the income tax as being 10% of the amount. If it is greater than 10000 then you compute the tax as being 10% of the first 10000 dollars and 15% of the amount over 10000 dollars.

```
      ....................
      ....................
      IF(AMOUNT.LT.0) GO TO 20
      ..... (here you should compute the tax and print the data and
             tax and go to the part of your program where you read
             more cards.)
      ....................
20    ........(here you want to print the data and the words
               'LOW INCOME' and go to the part of the program
               where you read more cards.)
      ....................
      ....................
       ....................
```

You can also read sections 4.2 through 4.5 for more detail. Notice that each time after reading a card you want to check the yearly salary for a negative value to stop the execution of the program.

OPTION II

You may choose either A or B below as your assignment. A is worth 8 points and B is worth 10 points.

PART A

OSWALD'S SWEATSHOP USES THE ANNUAL CHRISTMAS GIFT AS AN INCENTIVE FOR BETTER WORK ATTENDENCE. WORKERS WHO HAVE HAD 3 OR FEWER ABSENCES ARE GIVEN A SPECIAL STEAK DINNER. WORKERS WITH 4 OR MORE ABSENCES ARE PUBLICALLY AWARDED A 'BIG MAC'.

A DECK OF CARDS HAS BEEN PREPARED WITH ONE CARD FOR EACH WORKER. THE WORKER'S SOCIAL SECURITY NUMBER HAS BEEN PUNCHED IN COLUMNS 1-9 AND THE NUMBER OF ABSENCES HAS BEEN PUNCHED IN COLUMNS 10-12.

WRITE A PROGRAM WHICH WILL PRINT A LIST OF THE SOCIAL SECURITY NUMBERS AND BESIDE EACH WILL PRINT 'SPECIAL DINNER' OR 'BIG MAC' DEPENDING ON WHICH GIFT THAT WORKER IS TO RECEIVE. USE FORMATTED INPUT/OUTPUT IN YOUR PROGRAM. THE DATA TO BE USED IS ON FILE D4IIA. YOU MUST PROVIDE THE TRAILER CARD.

HERE IS A BIT OF HELP IF YOU NEED IT.
AFTER YOUR PROGRAM READS A CARD, YOU WILL WANT TO
TEST IT TO SEE IF IT IS THE TRAILER CARD.
(TRAILER CARDS ARE EXPLAINED IN SECT. 4.10 P95-97.)
SUPPOSE THAT YOU PUNCH -1 FOR THE VARIABLE ABSEN
IN YOUR TRAILER CARD.
AN APPROPRIATE STATEMENT WOULD BE 'IF(ABSEN.EQ.-1)
STOP'.

NOW YOU WANT YOUR PROGRAM TO USE ONE OF TWO SEGMENTS
OF CODE (I.E. GROUPS OF STATEMENTS)
DEPENDING ON THE NUMBER OF ABSENCES OF
THAT WORKER. THAT IS,
YOU WILL USE ONE SEGMENT FOR THOSE WHO GET SPECIAL DINNERS AND
ANOTHER SEGMENT FOR THOSE WHO GET BIG MACS. YOU
WILL NEED AN 'IF' STATEMENT TO CONTROL WHICH SEGMENT IS USED.
THUS YOU MIGHT HAVE 'IF(.....) GO TO 20'. ONE SEGMENT
WOULD THEN FOLLOW THIS 'IF' STATEMENT AND THE OTHER
WOULD START WITH A STATEMENT LABELED 20.
NOTE THAT BOTH SEGMENTS MUST, IN OUR CASE, END WITH A
TRANSFER OF CONTROL (GO TO) TO THE BEGINNING OF THE
PROGRAM TO READ THE NEXT CARD.
THE EXAMPLE IN THE MIDDLE OF PAGE 84 MIGHT HELP YOU.

PART B

OSWALD IS GIVING GIFTS BASED ON NUMBER OF ABSENCES AS
IN ASSIGNMENT A. NOW, HOWEVER, HE WANTS TO MAKE 3 CATEGORIES,
AND FIND THE TOTAL COST OF HIS GIFTS. ALSO, YOU MUST
IMPROVE YOUR OUTPUT.

TO WORKERS WITH NO ABSENCES HE WILL GIVE A $25.00 DINNER
(WOW!), TO THOSE WITH 1, 2, OR 3 ABSENCES HE WILL GIVE A $10.00
FROZEN TURKEY, AND TO THOSE WITH 4 OR MORE ABSENCES A
$1.50 RABBIT. YOUR OUTPUT SHOULD HAVE A COLUMN OF
SOCIAL SECURITY NUMBERS BESIDE WHICH IS A COLUMN
GIVING THE TYPE OF GIFT (DINNER, TURKEY, OR RABBIT) FOR
EACH WORKER, JUST AS IN A.

PRINT HEADINGS FOR YOUR COLUMNS. ALSO PRINT THE SOCIAL
SECURITY NUMBERS WITH DASHES IN THE USUAL PLACES, E.G.
555-44-3333. (HINT, USE 3 VARIABLES - ONE TO STORE
EACH PART OF THE NUMBER.)
BELOW THE COLUMNS SHOULD BE PRINTED THE TOTAL COST OF THE GIFTS.
THE DATA TO BE USED IS IN FILE D4IIA. YOU MUST
PROVIDE YOUR OWN TRAILER CARD.